The use of performability in the design of communication networks

by

Amardiya Sesmun

March 1997

A thesis submitted for the degree of Doctor of Philosophy
in the Faculty of Engineering of the University of London
and for the Diploma of Imperial College.



Digital Communications Section,
Department of Electrical and Electronic Engineering,
Imperial College of Science, College and Medicine,
Exhibition Road, London SW7 2BT.

BIBL. LONDIN. UNIV.

To Ma, Pa and Natacha

Abstract

Traditionally, computer and communication systems have been appraised with respect to either performance or reliability. The individual assessment of performance and reliability are difficult to combine so as to reflect a system's ability to perform over an observation period. This has led to the emergence of the concept of *performability*.

The main objective of the work reported on in this thesis is to investigate the use of performability in the design of communication networks, with the ultimate goal being to derive a design methodology for fault-tolerant networks. This calls firstly for the establishment of a performability evaluation technique. In the thesis, it is shown that the performability of a system, affected by faults and repairs, is related to its fault-tolerance. In view of these results, the concept of a *performability index* is proposed as a means of assessing the relative importance of individual components of a system with respect to their reliability and contribution to performance.

A three-stage design method is proposed whereby the first two stages are required to generate a network which satisfies an initial performance and basic reliability requirements. The *fault-tolerance augmentation stage* is then used to augment the network in order to meet performability requirements. A performability-based descent factor is used in the iterations of this stage. Results of the proposed design approach show that, although comparable in cost, the networks designed are much more fault-tolerant than networks generated using conventional methods. In some cases, due to a better placement of resources, a more fault-tolerant network may be obtained using fewer links.

The size of the network is a factor which seriously affects the computational complexity of performability-based design. A performability-based design method using the technique of clustering is proposed. The application of techniques of decomposition and aggregation to the network's performability model is investigated. Different scenarios as to how to approach the design are discussed.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor L. F. Turner for his guidance, support and encouragement throughout the project.

The work reported on in this thesis was supported by the BMDO under grant number N00014-93-1-0057 and managed by the U.S. Office of Naval Research, which I duly acknowledge. I also thank the Committee of Vice Chancellors and Principals of the Universities of the United Kingdom for offering me an Overseas Research Students Award.

I would like to thank Dr. J. Barria for several useful discussions and Dr. A. Manikas for his help, especially for pointing out the use of Mathematica to me. I would also like to extend my thanks to my colleagues of the Digital Communications Section who have provided me with a friendly environment to work in.

My stay in London would not have been so special and enjoyable without my friends. I would like to express my thanks to all my friends, especially Henza, Shamima and Pamela for all those happy moments and good laughs, Naushad for explaining the mysteries of matrices to me, Jenny for her care and constant encouragement, and Jacques for his moral support and also for being very patient with me.

Last but certainly not least, I am most indebted to my parents and my sister Natacha. My special thanks go to Natacha for taking good care of me, making sure I eat properly, and also for proof-reading my thesis. I cannot find enough words to thank my parents who, although miles away, have always been very understanding and encouraging. As a token of gratitude, however small it is, it is to them that I would like to dedicate this thesis with my love and affection.

Table of Contents

Abstract **3**

Acknowledgements **4**

Table of contents **5**

List of figures **13**

List of tables **17**

Chapter 1 - Introduction **18**

1.1 Introduction *18*

1.2 Problem formulation and objectives *19*

1.3 Organisation of thesis *23*

Chapter 2 - Review of the design of communication networks **27**

2.1 The design process: Introduction *27*

2.2 The design methodology *29*

2.3 Topological design *31*

2.3.1 Network topologies [SHA90] *32*

2.3.1.1 Mesh topology *32*

2.3.1.2 Star topology *33*

2.3.1.3 Bus topology *33*

2.3.1.4 Ring topology *34*

2.3.1.5 Tree or Multidrop (MD) topology *35*

2.3.1.6 Mixed topologies *36*

2.3.2 Synthesis of network topologies [SHA90] *37*

2.3.3 Computing the lower bound on the number of concentrators and switches [SHA90] *38*

2.3.4 Algorithms for locating concentrators and switches *39*

2.3.4.1 Problem formulation [SCH77] *39*

2.3.4.2 Add algorithm [SCH77] *40*
2.3.4.3 Drop algorithm [SCH77] *41*
2.3.4.4 Sharma's algorithm for locating concentrators and switches [SHA90] *42*
2.3.5 Algorithms for connecting CPE locations to concentrators and switches [SHA90] *45*
2.3.6 A generalised algorithm for designing all network topologies [SHA90] .. *45*
2.3.7 Design of a specific topology *46*
2.3.8 Algorithm for designing a multidrop (MD) topology *47*
2.3.8.1 Heuristic algorithms *47*
2.3.8.1.1 Prim's algorithm for designing an MST topology without capacity constraints [SHA90] *48*
2.3.8.1.2 Prim's algorithm for designing an MD topology with capacity constraints [SCH77] *49*
2.3.8.1.3 Kruskal's algorithm for designing an MD topology without capacity constraints *49*
2.3.8.1.4 Kruskal's algorithm for designing an MD topology with capacity constraints [SCH77] *50*
2.3.8.1.5 Esau-Williams algorithm for designing MD topologies with capacity constraints [SCH77] *50*
2.3.8.2 Kershenbaum and Chou's algorithm [KER74] [SCH77] *51*
2.3.8.3 Chandy and Russell's algorithm [CHA72] [SCH77] *52*
2.3.9 Distributed Network Design *54*
2.3.9.1 Cut Saturation algorithm [GER74] [SCH77] *54*
2.4 Capacity and flow assignment *56*
2.4.1 The capacity assignment problem [BER87] *56*
2.4.2 The flow assignment problem [KLE76] *58*
2.4.3 The capacity and flow assignment problem [KLE76] *60*
2.5 Fault-tolerant design - capacity augmentation *61*
2.5.1 Proofing method [NIS88] *62*
2.5.1.1 Max delay method [NIS88] *63*
2.5.1.2 Max average method [NIS88] *63*

2.5.2 Fault-tolerant optimal channel allocation and flow assignment problem [KAN92] 64

2.6 Conclusions 65

Chapter 3 - Performability : definition, model construction and solution **68**

3.1 Introduction 68

3.2 Definition of performability [MEY92] [MEY95] 72

3.3 Choice of the performance variable Y 73

3.4 Performability model construction and solution 73

3.5 Markov reward model 75

3.6 Review of techniques used to solve for performability 75

3.7 The randomisation technique [DES89] [DES92] 77

3.8 Use of randomisation to calculate transient performability measures [DES92] 79

3.9 Application of the randomisation technique 80

3.9.1 Modelling of failure/repair process in communication networks using the Markov reward model 80

3.9.2 Application of the randomisation technique to calculate the PDF of the average accumulated reward 85

3.9.3 Evaluation of the conditional distribution $P[ACR(t) \leq r \mid n, k]$ [DES92] 86

3.9.4 Evaluation of the conditional probability $\Omega(n, k)$ [DES92] 88

3.9.5 Results 89

3.9.5.1 Results A 90

3.9.5.2 Results B 91

3.9.6 Means of assessing the fault-tolerance of a system 93

3.10 Concept of performability index 94

3.10.1 Performability index : Definition 1 95

3.10.2 Performability index : Definition 2 97

3.10.3 Application of the performability index 99

3.11 Problems associated with Weisberg's result 106

3.12 Problems associated with the many-to-one mapping from performance to reward 108

3.13 Problems associated with the randomisation technique 109
3.14 Techniques of state-space reduction 110
3.15 Clustering or lumping of states of the Markov chain 112
3.16 Evaluating moments of performability [IYE86] 115
3.17 Conclusions 119

Chapter 4 - Proposal for a performability-based design method 123
4.1 Introduction 123
4.2 Design Proposal 128
4.2.1 Synopsis of design proposal 128
4.2.2 Assumptions 130
4.2.3 Initial topology generation stage 130
4.2.4 Connectivity establishment stage 131
4.2.5 Fault-tolerance augmentation stage 133
4.2.5.1 Choice of state 134
4.2.5.2 Concept of performability index 136
4.2.5.3 Augmentation strategy 137
4.2.5.4 General rules for fault-tolerance augmentation 138
4.2.5.5 Choice of topology 140
4.3 Definition of the Performability-based descent factor 141
4.4 Reward definition 142
4.4.1 Reward definition 1 143
4.4.2 Reward definition 2 143
4.4.3 Application of the two definitions in reward assignment 144
4.5 Definition of the cost function 146
4.5.1 Cost function: Definition 1 146
4.5.2 Cost function : Definition 2 147
4.6 Choice of performance measure and routing algorithm 147
4.6.1 Network simulation 148
4.6.2 Simulation model 148
4.6.3 Example 1: Coping with node failures 149

4.6.3.1 Network topology 149
4.6.3.2 Routing using Dijkstra's algorithm 150
4.6.3.3 Application of Dijkstra's algorithm in the simulations 151
4.6.3.4 Performance measure from simulation 152
4.6.3.5 Simulation results 154
4.6.3.6 Conclusion from example 1 156
4.6.4 Example 2: Case of use of non-optimal routing - probabilistic routing 156
4.6.4.1 Network topology 156
4.6.4.2 Essence of random routing 157
4.6.4.3 Simulation results 157
4.6.4.4 Conclusion from example 2 162
4.6.5 Optimal routing - GPM based [BER87] 162
4.7 Conclusions 166

Chapter 5 - Results of application of the performability-based design technique 170
5.1 Introduction 170
5.2 Comparison between use of the two cost functions in the descent factor 173
5.3 Application of the proposed design technique 175
5.3.1 Execution of Stage 1: initial topology generation stage 176
5.3.2 Execution of Stage 2: connectivity establishment stage 177
5.3.3 Execution of Stage 3: fault-tolerance augmentation stage 179
5.3.4 Application of fault-tolerance augmentation 181
5.4 Set A - Capacity augmentation based design 182
5.5 Set B - Increase in connectivity based design 186
5.6 Set C - Comparison at different numbers of iterations 189
5.7 Set D - Comparison between Tan and Kang's approach and a joint performance/ performability - based approach 192
5.8 Comparison of the design approaches in terms of performance 196
5.9 Conclusions from comparing the different design approaches 197
5.10 Comparison between adding a link and augmenting capacity of an existing link 201

5.11 Set E - Comparison between increase in connectivity and capacity augmentation 205
5.11.1 Pure performance-based design 205
5.11.2 Joint performance / performability - based design 207
5.11.3 Pure performability-based design 209
5.12 Design using both mechanisms 211
5.13 Set F - Intermediate states of failures 214
5.14 Conclusions 217

Chapter 6 - Performability-based design of clusters **221**
6.1 Introduction 221
6.2 Clustering of nodes 223
6.3 Connecting the nodes 224
6.4 Design of clusters after establishment of inter-cluster links 224
6.5 Design of clusters prior to establishment of inter-cluster links 225
6.5.1 Worst-case scenario 226
6.5.2 Best-case n scenario 226
6.5.3 Best-case 1 scenario 227
6.6 Modelling issues 229
6.7 Review of existing techniques 230
6.7.1 Aggregation of large Markov Chains [SCH90] 230
6.7.2 Theory of decomposition 232
6.7.3 Mathematical description of NCD [COU77] 232
6.7.4 The process of aggregation 235
6.7.5 Aggregation in the steady state [COU77] 236
6.8 Application of decomposition and aggregation to the network model 238
6.8.1 Description of the current network model 238
6.8.2 Relation between the overall network and clusters' reliability models .. 240
6.8.3 Relation between the overall network and the clusters' performance model 246
6.8.4 Design of clusters 246

6.8.5 Aggregation of the network model under steady-state conditions..... 247
6.8.6 Proposed aggregation technique for transient analysis 247
6.8.6.1 Aggregation for the completely decomposable network..... 247
6.8.6.2 Aggregation for the nearly completely decomposable system.. 248
6.8.7 Validation against the actual overall network model..... 249
6.8.8 Establishment of a stopping criterion for design..... 250
6.8.9 Error analysis..... 250
6.9 Results 252
6.9.1 Example 1: design under steady-state conditions..... 252
6.9.2 Example 2 - design under transient conditions..... 254
6.9.3 Example 3: worst-case scenario 257
6.9.4 Example 4: best-case 1 scenario..... 258
6.9.5 Example 5: best-case n scenario..... 259
6.9.6 Comparison between the scenarios..... 259
6.10 Conclusions 261

Chapter 7 - Conclusions and the way ahead..... 266
7.1 Conclusions 266
7.2 The way ahead..... 272

Appendices 274
AppendixA1 275
AppendixA2 277
AppendixA3 278
AppendixA4 279
AppendixB1 281
AppendixB2 283
AppendixB3 285
AppendixB4 287
AppendixC1 289

AppendixC2 ..... *290*
AppendixC3 ..... *292*
AppendixD1 ..... *295*
AppendixD2 ..... *297*
AppendixD3 ..... *299*

References **301**

List of figures

Figure 2.1 Relationship between modelling, analysis and design [SHA90] 29

Figure 2.2 Network modelling 30

Figure 2.3 Iterative network design methodology 31

Figure 2.4 The mesh topology 32

Figure 2.5 The star topology 33

Figure 2.6 The bus topology 34

Figure 2.7 The ring topology 35

Figure 2.8 The multidrop topology 36

Figure 2.9(a) Locating one centre of gravity 43

Figure 2.9(b) Locating 2 new centres of gravity 43

Figure 2.9(c) Locating 4 new centres of gravity 44

Figure 3.1 Mapping from performance on to reward 82

Figure 3.2 2-node network 82

Figure 3.3 Model for 2-node network 83

Figure 3.4 Model for a single link 83

Figure 3.5 General network model 84

Figure 3.6 Results when rates of structural changes are varied 90

Figure 3.7 Results when performance data is varied 92

Figure 3.8 Typical sample path for a system of three components 96

Figure 3.9(a) Sample path for component 1 for Definition 1 96

Figure 3.9(b) Sample path for component 2 for Definition 1 96

Figure 3.9(c) Sample path for component 3 for Definition 1 97

Figure 3.10(a) Sample path for component 1 for Definition 2 98

Figure 3.10(b) Sample path for component 2 for Definition 2 98

Figure 3.10(c) Sample path for component 3 for Definition 2 98

Figure 3.11 Typical distribution of indices 99

Figure 3.12 System of three processors 100

Figure 3.13 Index in range [0, 0.5] according to Definition 1 101

Figure 3.14 Index in range [0.6, 0.9] according to Definition 1 101

Figure 3.15 Index in range [0, 0.5] according to Definition 2 103

Figure 3.16 Index in range [0.6, 0.9] according to Definition 2 103

Figure 3.17 Performability distribution in range [0, 0.5] 105

Figure 3.18 Performability distribution in range [0.6, 0.8] 105

Figure 3.19 Markov chain of three states 114

Figure 4.1 Schematic representation of the overall design method..... 128

Figure 4.2(a) Minimal Spanning Tree connecting nodes 131

Figure 4.2(b) Network generated after Pass One of the increase in connectivity stage.. 132

Figure 4.2(c) Network generated after Pass Two of the increase in connectivity stage..132

Figure 4.3 Scenario for application of rule 1 139

Figure 4.4 Scenario for application of rule 2 139

Figure 4.5 Scenario for application of rule 3 140

Figure 4.6 Scenario for application of rule 5 140

Figure 4.7 Typical profiles for performability distributions after n and $(n + 1)$ iterations 142

Figure 4.8 Configuration of a 6-node network 149

Figure 4.9 Configuration of a symmetrical 6-node network 157

Figure 4.10 Distribution of traffic at a source node 159

Figure 4.11(a) Comparison between fully operational state and configuration where primary link has failed over range x between [0.2, 0.3] 159

Figure 4.11(b) Comparison between fully operational state and configuration where primary link has failed over range x between [0.3, 0.5] 159

Figure 4.12 Comparison between network in a fully operational state and in a configuration where a secondary link has failed 160

Figure 4.13 Comparison between network in a fully operational state and in a configuration where a tertiary link has failed 161

Figure 5.1 6-node ring network 173

Figure 5.2 Minimal Spanning Tree based on physical distance 176

Figure 5.3 Minimal Spanning Tree based on traffic demands 177

Figure 5.4 Network designed using a distance criterion at Stage 2 177

Figure 5.5 Network designed based on traffic demands at Stage 2 178

Figure 5.6 Performability distributions of networks designed using distance and traffic demands as criterion 179
Figure 5.7 Point-performability per iteration of design of networks initially based on distance and demands, using increase in connectivity 181
Figure 5.8 Point performability per iteration of design of networks initially based on distance and demands, using capacity augmentation 182
Figure 5.9 Performability distributions of the networks designed using capacity augmentation 185
Figure 5.10 Performability distributions of the networks designed using increase in connectivity 187
Figure 5.11 Performability distributions of the networks designed using increase in connectivity over reward range [0.4, 0.9] 188
Figure 5.12 Performability distributions of the networks after 5 iterations of an increase in connectivity-based approach 189
Figure 5.13 Point-performabilities of networks designed using capacity augmentation after every three iterations 190
Figure 5.14 Point-performabilities of networks designed using increase in connectivity at each iteration 190
Figure 5.15 Point-performabilities of networks designed using increase in connectivity when cost is considered 191
Figure 5.16(a) Comparison between Kang's and the performability-based approach after the 1st iteration 193
Figure 5.16(b) Comparison between Kang's and the performability-based approach after the 2nd iteration 193
Figure 5.16(c) Comparison between Kang's and the performability-based approach after the 3rd iteration 194
Figure 5.16(d) Comparison between Kang's and the performability-based approach after the 4th iteration 194
Figure 5.17 Point-performabilities of networks designed using Tan and Kang's and the performability-based approaches per iteration 195
Figure 5.18 Performance of networks designed using increase in connectivity, in the fully operational state 196

Figure 5.19 Performance of networks designed using capacity augmentation, in the fully operational state 197

Figure 5.20 Sample path for a network 198

Figure 5.21 Sample paths of network after a capacity augmentation 199

Figure 5.22 Performance of networks designed using each mechanism according to a pure performance approach 206

Figure 5.23 Performability distributions of networks designed using each mechanism according to a pure performance approach 207

Figure 5.24 Point-performabilities of networks designed using each mechanism according to the joint performance / performability approach 208

Figure 5.25 Performability distributions of networks designed using each mechanism according to a joint performance / performability approach ..209

Figure 5.26 Point-performabilities of networks designed using each mechanism according to the pure performability approach 210

Figure 5.27 Performability distributions of networks designed using each mechanism according to the pure performability approach 210

Figure 5.28 Markov chain model for a single link 215

Figure 5.29 Performability distributions of networks when intermediate states of failure are considered 216

Figure 5.30(a) Point-performability of networks for iterations 0 to 2 216

Figure 5.30(b) Point-performability of networks for iterations 3 to 6 216

Figure 6.1 Clusters of nodes 225

Figure 6.2 Network configuration in the *worst-case* scenario 226

Figure 6.3 Network configuration in the *best-case n* scenario 227

Figure 6.4 Network configuration in the *best-case* 1 scenario 228

Figure 6.5 Overall network model 241

Figure 6.6(a) MRM for cluster 1 241

Figure 6.6(b) MRM for cluster 2 241

Figure 6.7 Original network model 242

Figure 6.8 Expanded network model 243

List of Tables

Table 2.1 Table of weights 52

Table 3.1 Performability indices assuming all repair rates are equal 101

Table 3.2 Performability indices for different mean repair times 102

Table 4.1 Reward definitions for Network1 and Network2 145

Table 4.2 Initial performance of the 6-node network 153

Table 4.3 Performance of the 6-node network when a maximum delay is imposed..... 155

Table 4.4 Comparison between the GPM-based routing algorithm and Dijkstra's algorithm for the network in Figure 4.8 165

Table 4.5 Comparison between the GPM-based routing algorithm and probabilistic routing for the network in Figure 4.9 166

Table 5.1 Cost and performability of network 174

Table 5.2 Performance levels of the networks designed when capacity augmentation is used 184

Table 5.3 Order of application of augmentation mechanisms to network in Figure 5.4 213

Table 6.1 Comparison of means of the clusters, aggregated, and whole network ..255

Table 6.2 Table of results for cluster 1 255

Table 6.3 Table of results for cluster 2 256

Table 6.4 Comparison of errors involved in aggregation 256

Table 6.5 Comparison of lower bounds on expected means and actual mean 257

Table 6.6 Comparison of cluster 1 designed according to the three scenarios in terms of capacity 260

Table 6.7 Comparison of the performance of cluster 1 designed according to the three scenarios.....260

Table 6.8 Comparison of the performance of the network designed according to the different scenarios 261

Chapter 1

Introduction

1.1 Introduction

In the world of today, society is becoming increasingly reliant on computer and communication networks, be it in industry, banking, academic institutions or even at home. Whether it is in the form of a Local Area Network connecting a number of PCs in an office, or a Wide Area Network spanning different cities, the *network* is a concept that has evolved and whose benefits are being increasingly exploited.

First and foremost, a network, whether a telephone network, a data network or an integrated network, is a means of connecting people or users and allowing communication between them. Initial applications of computer networks centred on remote accessing of central storage facilities, for example a common file server and databases. Remote accessing and updating of databases are required by a number of applications such as airline reservation systems, automatic teller machines, inventory control systems, automated order entry systems or even weather tracking systems. Traditional services offered to users also include electronic mail, file transfer protocol and telnet, enabling login to remote hosts. With the advent of modern services like the world-wide web, computer and communication networks are allowing the dissemination of a vast amount of information to a large number of outlets all over the world. The more the computer network is used as a sophisticated communication system, the more the limitations of physical distance and travel are overcome. Techniques of video conferencing are but an example.

With the ever increasing demand for and dependence upon computer and communication networks, there is an even greater need for efficient and reliable systems. Not only do the systems have to be robust to avoid frequent failures, but they also must be able to deliver high quality service. More importantly, they have to be capable of

maintaining their quality of service or performance even in the presence of component failure.

In other words, a fault-tolerant computer or communication network is highly desirable. Such a system is one where the occurrence of a fault, software or hardware, although reducing the quality of a delivered service, leaves the level of service at or above some pre-specified acceptable level. The need to design fault-tolerant communication networks constitutes the motivation behind the work described in this thesis. Traditional design methods are studied and a means of evaluating fault-tolerance is investigated, culminating into a proposal for a design methodology for fault-tolerant networks.

1.2 Problem formulation and objectives

Efficient design and synthesis of communication networks constitutes an extremely complex problem because of the number of variables, requirements and constraints which are involved. Requirements are usually specified in terms of performance and reliability, within constraints imposed by a budget. The variables refer to the topological configuration, channel capacity assignment, routing procedures, flow control procedures and queueing discipline. The goal of optimal network design and facility engineering is to generate network topologies that minimise total network cost while selecting facility types, allocating capacity and routing traffic to accommodate demand and meet performance and reliability requirements [GER90]. In addition, to obviate loss of network services, communication networks should be designed so that they remain operational and maintain as high a performance level as possible, or is needed, even in the presence of network component failure.

Owing to the complexity of the design problem, Gerla and Kleinrock [GER77] have decomposed it into three non-linear sub problems: optimal topological design, optimal capacity assignment and optimal routing. The topological design problem is in itself very difficult; the structure of the network is on one hand affected by the nature of the cost function and on the other hand determines its behaviour in terms of performance and reliability. Once the topology of the network has been determined, each channel needs to be

assigned a certain capacity, sufficient for the network to meet performance requirements. Furthermore, the routing algorithm to be adopted and the queueing discipline that governs the order of service for the various channel queues have to be decided upon. Capacity assignment and type of routing have a greater bearing on network performance rather than on reliability.

Traditionally, design of communication networks has been done with respect to performance and reliability separately, although they are closely related. In topological design, for example, the cut saturation method [GER74], reliability is specified as a 2-connectivity criterion; performance is then optimised, assuming that no component failures occur. Capacity assignment [BER87] is usually done in the state where all components are functional, and optimises performance, with no regard for the reliability of the elements constituting the network. Capacity augmentation techniques have been proposed by Nishida and Miyahara [NIS88] and Kang and Tan [KAN92], which improve performance in states of failure. Although Nishida and Miyahara [NIS88] attribute a probability to each of the configurations in which the network can be, which is determined by the rates of failing of the components, no account is taken of the mean repair time. Kang and Tan [KAN92] also treat performance and reliability issues separately.

Even though they are closely related concepts, as indicated above, performance and reliability have been investigated separately. System reliability addresses questions such as what is the probability that the system fails given that it starts off in a fully operational state, or what is the probability that there is successful communication between any pair of nodes. For example, if reliability is expressed in terms of connectivity, the network may be considered to have failed if any disconnection occurs as a result of a link failing. While the network remains connected, it is judged to be operational. Reliability analysis is binary-valued with respect to performance in the sense that performance is either good or bad; performance is good when the network is operational and bad when it has failed. There is no consideration given to the different levels of performance which may be attained if links fail and do not necessarily cause the network to be disconnected. Therefore, a design approach based only on reliability considerations does not necessarily avoid performance degradation, even if the network is guaranteed to be connected during the failure of the network element.

On the other hand, performance is a measure of how efficiently a system delivers a specified service. Typically, performance is expressed as the throughput of a system, the average packet delay across a network, or the queue length at a server. Generally, performance modelling tools, whether they use simulation or analytical methods, consider the system under analysis to be in a particular state, or configuration, and assume that it remains in that state during the whole observation period; no structural changes are allowed. In this case, there is no regard for how often components fail and their effect on the performance of the overall system. Therefore, a design approach where network performance is optimised without considering network failures can yield a network where performance degrades drastically on the occurrence of a component failure. As a result, neither a pure reliability approach, nor a performance-oriented approach is enough to design a reliable and fault-tolerant network.

A fault-tolerant network can be referred to as a gracefully degrading system, whereby performance does not degrade drastically upon component failure. For such a system, individual assessment of performance and reliability are not easy to combine so as to capture this property of being able to withstand failures. The need for a measure which characterises and evaluates this type of system has led to the emergence of the concept of *performability* from a synergy of performance and reliability. The concept was introduced by Meyer. A general modelling framework that permits the definition, formulation, and evaluation of performability was published in 1980 [MEY80]; previously, Meyer [MEY78] had given a less refined description.

Performability evaluation considers the system under investigation for a given observation period, and assesses its *ability to perform* in a designated environment, taking into account the fact that structural changes occur which affect performance during that interval. As distinct from reliability investigation, the system's operational integrity is no longer viewed as a binary variable, that is either "up" or "down". The system's operational performance is considered as a multi-valued variable which changes according to the extent to which the system is faulty. Also, assessment is not constrained to a particular state, as in performance evaluation, but considers the system as it evolves through different states during the observation period.

Most of the research in the field of performability has been in the area of model specification, construction and solution. Model specification and construction involve deciding on the performability measures of interest, choosing a stochastic process to characterise the system, that is the base model, and relating the base model to the measures such that a solution can be derived. Generally, solution methods use some form of rate-based Markov Reward Model, where the underlying Markov process is augmented by a reward structure. A review of these solution techniques is given in Chapter 3.

As far as the application of performability is concerned, most of the research has centred on computer-based systems. On the hardware side, operational faults have been of main concern, including degradable microcomputers [CON92] [CON93], random access memories [CIC92], disk arrays, fault-tolerant multiprocessor systems [AUP91] [GRA91] [LOP93], as well as fault-tolerant real-time systems [ISL91] [ISL92]. Issues regarding structured software fault-tolerance and various schemes for enhancing the performability of fault-tolerant software have been addressed [TAI93]. Performability has been used in the evaluation of CSMA/CD protocols [PRO93]. The concept of performability has also been applied to distributed systems [CAS94] [FRA94], manufacturing systems [VIS94] [NAR94] and more recently to distributed database systems [TAI96].

The objective of the present research translates into an investigation of the use of performability in the design of networks, with the ultimate goal being to derive a design methodology for fault-tolerant networks. The communication network aimed at is one whose performance does not degrade drastically upon component failure. The questions that need to be addressed are how to approach the design problem, how to evaluate the performability of a network, and finally how to apply performability to design of communication networks. In so doing, any problems which may arise from computational issues need to be tackled.

Design being so complex, the problem of finding a globally optimal solution taking into account all the design variables and constraints imposed, appears almost intractable. The approach adopted is to tackle the three sub-problems identified by Gerla and Kleinrock separately. The first stage consists of generating a network topology which meets basic

reliability requirements; the problems of topological design and capacity assignment are solved separately. An optimal routing algorithm can then be applied to determine the performance of the network in all the states in which it can be, as dictated by failures of its components. Techniques for the incorporation of redundancy and capacity augmentation are used in an attempt to improve the network's ability to withstand failures and their effects. This process of augmenting the fault-tolerance of the network is performability-based. Having a performability-based design approach calls for the need to implement a performability evaluation technique prior to development of the design methodology.

1.3 Organisation of thesis

This thesis is organised in the following way: firstly, the design problem, its complexity and how it can be approached are discussed in Chapter 2; then the concept of performability is introduced, and techniques to construct a suitable model and solve for it are described in Chapter 3. Design and performability evaluation are merged in Chapter 4 to generate the performability-based design method. Results of application of the proposed performability-based design technique are presented in Chapter 5. The size of the network affects the computation involved in performability-based design. In Chapter 6, a clustered-based approach to design is investigated in an attempt to overcome the problems arising as the size of the network increases.

In more detail, in Chapter 2, the overall design problem is discussed, with particular emphasis on topological design and capacity assignment. A number of topologies, which have proved to be useful in communication networks, are described. The various steps involved in topological design are explained with algorithms given for each of them; all these steps are not used in the proposed design methodology in this thesis, but are included for completeness and to give an insight into the complexity of design. The individual and joint capacity and flow assignment problems are also described; capacity augmentation techniques, used as means of enhancing the fault-tolerance of networks, are reviewed. Description of the various algorithms will show how performance and reliability are treated separately in the design of communication networks. As already explained, if a network exhibits degradable performance, that is, a component failure causes performance to be

reduced without entailing failure of the system, neither a pure performance, nor a pure reliability analysis can detect this behaviour. Performance analysis assumes that all components of the network are operational whereas reliability analysis views the network as either operational or not, irrespective of the level of performance achieved while it is functional.

This calls for the need to use a composite measure of performance and reliability, which is performability, in the design of communication networks. In Chapter 3, the concepts of performance, reliability and performability are first explained. Performability is defined mathematically, and a suitable model for its evaluation, the Markov Reward Model, is constructed. This model consists of a Markov process which represents the evolution of the network as it is affected by structural changes. It is augmented by a reward structure which can give an indication of the performance of the network as it changes states. A number of solution methods are reviewed; the one chosen for performability evaluation in this thesis is the *randomisation technique*. The reasons for the choice are its simplicity of implementation and the fact that it can be adapted easily to calculate various measures from a distribution of average accumulated reward to point availability and reliability [DES89] [DES92]. An investigation of how performance and reliability data affect the performability of a system leads to the statement that performability, evaluated as a distribution of average accumulated reward, is related to its fault-tolerance. In view of these results, the concept of a *performability index* is proposed as a means of assessing the relative importance of individual components of a system with respect to their failure rates, mean repair times and contribution to performance of the overall system. The degree of computation required in performability evaluation is directly dependent on the size of the state-space. Techniques of state-space reduction are reviewed, with the *lumping technique* being described in more detail. Results of coupling the randomisation technique with lumping are given. Moments of performability constitute interesting measures; an algorithm by Iyer *et al.* [IYE86] is given as a means of evaluating these measures.

Having established a means of evaluating performability, it can now be applied to the design of communication networks. In Chapter 4, a new three-stage design strategy is proposed which is performability-based. The first two stages are required to generate a

network which satisfies an initial performance and basic reliability requirements. The most important stage, the *fault-tolerance augmentation* stage is then used to augment the network, via the addition of more links, or through the augmentation of the capacity of existing links, in order to meet performability requirements. The difference between the proposed design technique and conventional design methods resides in the approach to augmenting the network. Whereas, traditionally, any optimisation is applied to the network, assuming that it is failure-free, in order to obtain the best performance under such operating conditions, the performability-based design method firstly applies augmentation in a state of failure, and secondly does so to design the network which performs best over a period of time, taking into consideration occurrence of structural changes such as failures and repairs. This stage is iterative and a performability-based descent factor is used to select the best network at the end of each iteration. Evaluation of performability requires computation of the performance of the network in the states in which it can be considered. Issues regarding choice of the performance variable and routing are addressed. The performability-based descent factor also includes the aspect of cost. The cost of a network is evaluated in terms of the cost of its constituent components. Two definitions of cost are considered where both involve the cost term just described, but the second definition also includes the average repair cost that is expected to be incurred over a period of time. The rationale behind proposing the second definition is that given that performability evaluation considers the network over a period of time when components fail and need to be repaired, it is only reasonable to take into account the cost that is expected to be incurred when such changes occur.

Chapter 5 is devoted to a presentation and discussion of results of the application of performability to design of communication networks. The capability of the proposed design strategy, its relative merits and shortcomings *vis-à-vis* conventional design techniques are established. Results show that, although comparable in cost, the networks designed are much more fault-tolerant than the ones generated using traditional design methods, which treat performance and reliability separately. In some cases, due to a better placement of resources, a more fault-tolerant network can be obtained using less resources, which has a significant implication with respect to the cost of the network. Furthermore, results show that such an approach to design is not necessarily detrimental to the rate at which

performance is improved in the fully operational state. The price paid for fault-tolerant design is in terms of the computational requirements of the method. Performance has to be evaluated for all the states in which the network is allowed to be, as dictated by failures and repair of its components in order for performability to be determined. As the size of the network grows, the number of states to be considered increases, thereby contributing to increasing the computational and memory requirements of the performability evaluation technique, and the time involved in the design process itself.

The size of the network is, therefore, a factor that has a direct impact on the complexity of a performability-based design. In Chapter 6, this issue is addressed. In order to reduce the complexity, an approach is considered in which the network is grouped into a number of inter-connected clusters, and each cluster is designed separately. Such an approach gives rise to a number of questions at both the design and modelling stages. In terms of design, there are different ways of handling intra-cluster traffic and inter-cluster traffic, which lead to derivation of a number of different scenarios. The relative merits and shortcomings of these approaches are discussed. From a modelling point of view, the problem is viewed as a case where a system is subjected to the processes of decomposition and aggregation. The theory of decomposition and aggregation as explained by Courtois [COU77] is reviewed. Decomposition of a performability model requires decomposition of the reliability as well as the performance model. Consideration of how to decompose the reliability model leads to proposal of a transformation which needs to be applied to the model in order to be able to relate the overall model to its constituent sub-models. Two aggregation mechanisms are proposed for generating an estimate of the overall result associated with the network using values obtained for the individual clusters. The issue of how to bound the error involved in this process is also discussed. Results are then presented when the network of ten nodes is decomposed into two clusters, and design is carried out under steady-state and transient conditions. Examples of how the performance of the network varies in the different scenarios occurring in consideration of inter-cluster traffic are also given.

In Chapter 7, conclusions derived from the research are presented and suggestions for future work are given.

Chapter 2

Review of the design of communication networks

2.1. The design process: Introduction

Efficient design of communication networks is far from an easy task due to the large number of variables and constraints that need to be taken into consideration. The variables relate to the topological configuration, the assignment of capacity to channels, the routing strategy, the flow control procedure and the queueing discipline adopted. These factors further depend on user requirements, or traffic demands. While performance and reliability represent the main requirements, cost acts as a major constraint. Given the large dimension of the design problem, Gerla and Kleinrock have decomposed it into three sub-problems, each of which can be solved separately. These are topological design, capacity assignment and the determination of the routing mechanism [GER77].

Topological design involves finding the best locations for switching nodes and connecting them in such a way as to meet performance and reliability requirements. Capacity assignment consists of allocating transmission capacity to the links in order to satisfy the traffic demands and performance criteria. Routing, a quite difficult problem in itself, addresses the issues of which links should carry information from one node to another so as to optimise performance. When solving all of these three problems, possibilities of failure have to be taken into account and any solution must attempt to cater for recovery in case of breakdowns in the network. As far as topological design is concerned, this is usually expressed in terms of a connectivity requirement which ensures that the network does not become disconnected under conditions of link failures. For instance, if a 2-connected criterion holds, then the network remains connected in the presence of any single link failure. In capacity assignment, additional capacity is allocated to each link so that it can accommodate any extra traffic arising when other components fail. Adaptive routing copes with failures by finding an alternative path for traffic upon occurrence of such structural changes.

Given the large number of parameters and constraints, it is not difficult to appreciate the increase in complexity of the design problem with a growth in network size. As the number of nodes to be connected increases, the number of potential links increases exponentially; how much capacity to assign to the links and what is the best routing strategy become more difficult questions to answer. In order to ease the problem, traditionally, designers have viewed a network as a hierarchical system. At the lowest level, there are the customer premise equipment and terminals which need to be connected to concentrators. The concentrators may then link to a central processing facility as in centralised design, or be interconnected as in distributed design. They may also be connected to more powerful concentrators at a higher level, which are themselves then interconnected. By considering the network as such, the overall design problem can be split into a number of sub-problems, whose solutions can be combined. Design of how the customer premise equipment is linked to the concentrators can be done, irrespective of how the concentrators are actually interconnected. Furthermore, determination of the topology which connects the concentrators need not take into consideration the way in which they are linked to terminals or customer premise equipment; only the amount of traffic from / to them has to be known.

In this thesis, the problem addressed focuses on the interconnection of nodes at a concentrator level and ignores the aspect of connecting terminals or CPE to concentrators. However, in this chapter, a review of the overall design problem is given. Firstly, the iterative way in which design is conducted is described in Section 2.2. The sub-problems of topological design and capacity assignment are dealt with in the remaining sections. The routing problem is considered only as a flow assignment problem. The use of different routing strategies is explained in Chapter 4. As far as topological design is concerned, there are a number of topologies which have proved to be useful in the field of communication networks and a description of the different topologies is given in Section 2.3.1. The steps involved in synthesis of networks are discussed in Section 2.3.2. The main steps consist of computing the number of concentrators and switches required, finding suitable locations for them, connecting the CPE to concentrators, and then interconnecting the concentrators or switches. The question of computing the number of concentrators and switches needed is considered in Section 2.3.3. In Section 2.3.4, a discussion of algorithms for locating concentrators and switches is given. Three algorithms, the *add*, the *drop* and an algorithm

due to Sharma are described for solving this problem. The process of connecting CPE locations to concentrators and switches is explained in Section 2.3.5. The steps in the design of a general topology are summarised in Section 2.3.6. The design of a specific topology is treated in Section 2.3.7, with Section 2.3.8 being devoted, in particular, to design of the multidrop topology; this is dealt with as a case of a centralised design problem where a number of terminals need to be connected to a central facility. A number of algorithms applicable to this topology are outlined. Distributed network design is then discussed in Section 2.3.9, using the cut saturation algorithm. The capacity and flow assignment problems are tackled in Section 2.4, with algorithms being given for the individual as well as the joint problems. In Section 2.5, methods of using capacity augmentation as a means of fault-tolerant design are described: the proofing method [NIS88] is described in Section 2.5.1, and an approach taken by Kang and Tan [KAN92] is given in Section 2.5.2. The chapter is concluded in Section 2.6.

2.2 The design methodology

The difficult task of producing an acceptable network design, taking into account all parameters and constraints, is certainly not a one-time effort. As illustrated in Figure 2.1, it is an iterative process which combines modelling, design, and analysis.

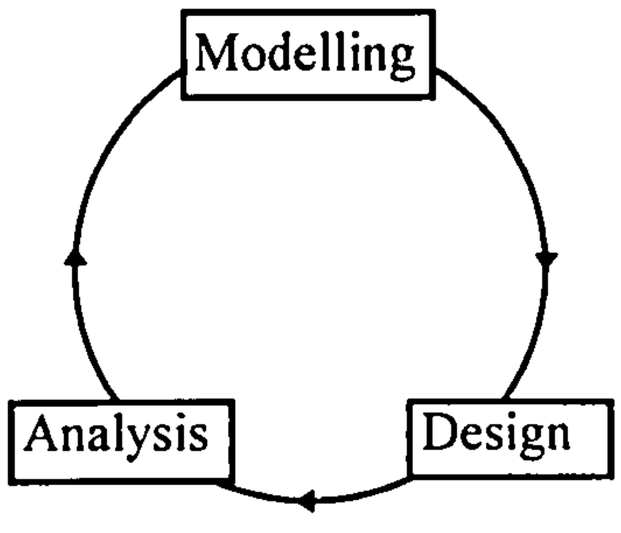
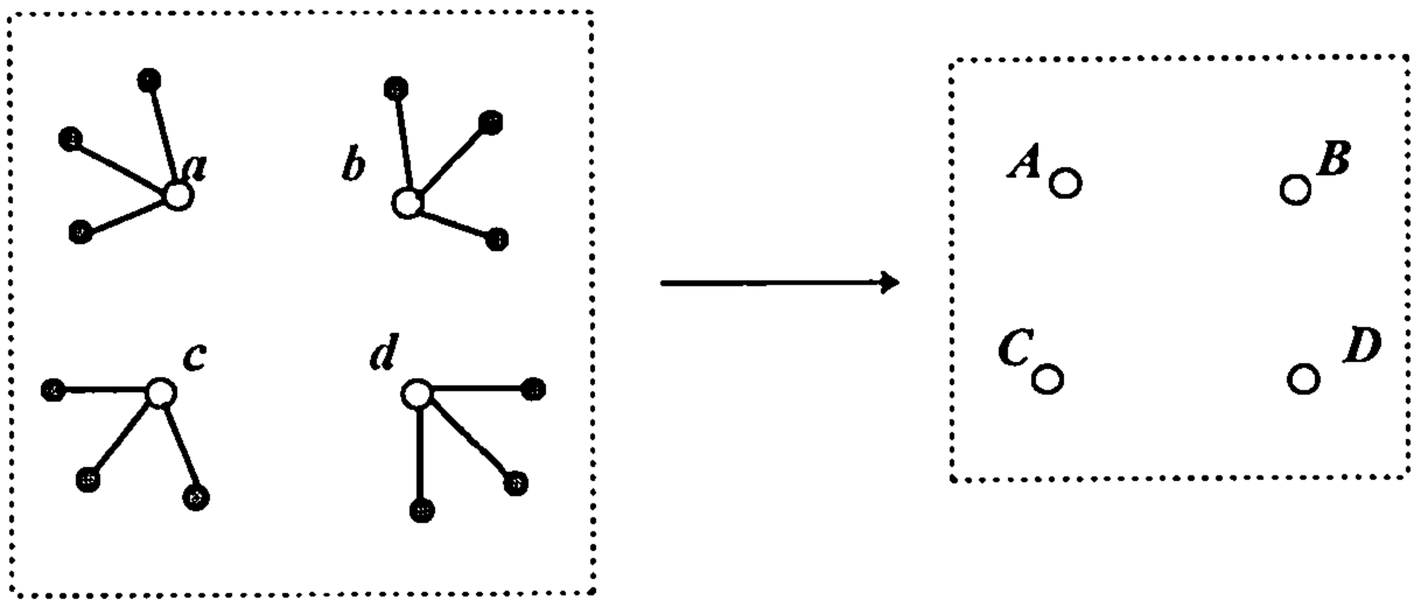


Figure 2.1 Relationship between modelling,
analysis and design [SHA90]

Network modelling can be done at any level of detail. The entire system can be treated as it is, or certain groups of nodes can be abstracted as a single node. The example in Figure 2.2(a) illustrates this.



(a) (b)

Figure 2.2 - Network modelling

Assume that the design problem consists of deriving the best way of connecting nodes ***a***, ***b***, ***c*** and ***d***, which could be concentrators. The entire system can be modelled as Figure 2.2(b), where for instance, node ***a*** and the grey coloured nodes, (●), connected to it are represented by a single node ***A***. The grey coloured nodes could correspond to terminals.

Having derived a model of the network, a design effort yields a network system architecture and topology, which is expected to meet the desired performance requirements. It involves mainly the generation of a suitable topology, capacity assignment, and choice of the routing strategy. Then, analysis of the system provides a prediction of its performance. Evaluation of performance can be done in a number of ways: through actual measurements of the physical system, via computer simulations, or using analytical techniques. The relative merits and drawbacks of these methods are given in Chapter 3.

The iterations of design proceed according to Figure 2.3. Once a feasible design is obtained, analysis techniques are applied to evaluate its performance. If the design does not meet specifications, design parameters such as capacity, or configuration, need to be altered to produce a new network. This configuration is subjected to analysis again, and these steps are repeated until an acceptable design is produced. Then the process stops and the design is documented.

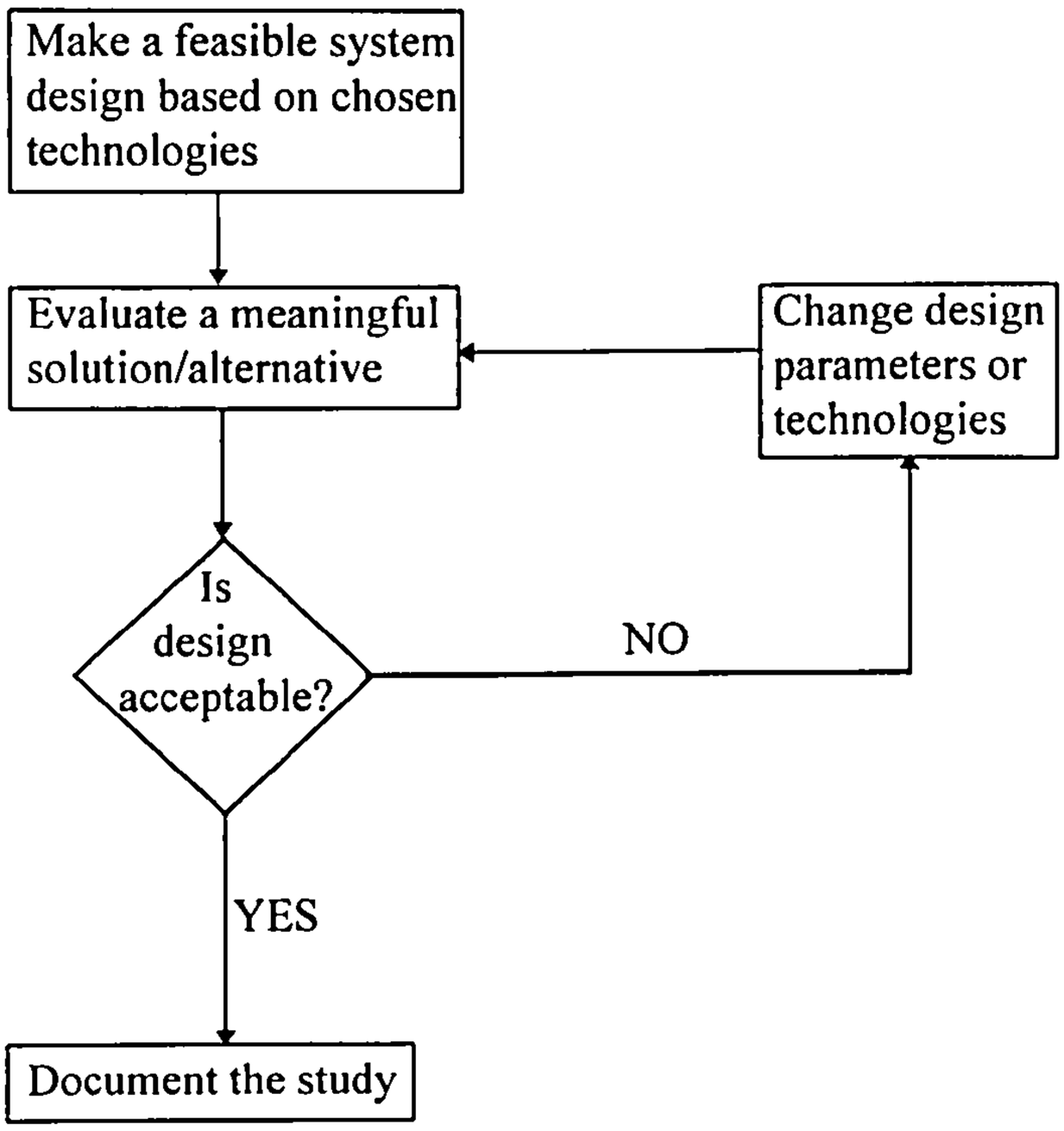


Figure 2.3 - Iterative network design methodology

Generating an acceptable design consists mainly of solving the sub-problems of topological design, capacity assignment and routing. The issues involved in the first two sub-problems are discussed in the following sections.

2.3 Topological design

The overall topological design problem consists of deciding where, in a geographically-distributed network, to locate concentrators or multiplexers, how many to use, which terminals to associate with which concentrators, and finally how to interconnect the concentrators. This has to be done in such a way as to achieve a configuration which meets performance and reliability requirements and satisfies constraints imposed by cost. A number of topologies have proved to be particularly suitable at the different levels of hierarchy in telecommunications networking; a brief discussion of the different topologies is given first.

2.3.1 Network topologies [SHA90]

2.3.1.1 Mesh topology



(a) Fully connected mesh (b) Partially connected mesh

Figure 2.4 -The mesh topology

A fully connected mesh topology provides a direct link or path between every pair of network nodes at a given level of hierarchy; the communication paths in a fully connected mesh are reduced to a single-hop variety. Such a topology is associated with a serious drawback in terms of the number of links required. For a network of n nodes, $n(n - 1)/2$ bidirectional links are necessary to ensure a fully connected network. From an economic point of view, this type of topology is feasible only when the number of nodes at a given network hierarchy is small. However, a mesh topology is associated with a high reliability in the sense that on the occurrence of any link failure between two nodes, there exists an alternative path for communication to proceed. Given that the cost of transmission in most existing networks is predominantly influenced by link costs and not switching, a fully connected mesh topology is suitable only at a very high level of hierarchy, where relatively few nodes need to be interconnected and a high degree of connectivity is required to ensure reliability. A partially connected mesh topology is generally recommended when the traffic flow between certain nodes is negligible and it can be switched via longer paths.

2.3.1.2 Star topology

A star topology consists of a single switching node, or hub node, and a number of subscriber nodes directly connected to it. Any node can communicate with any other node via this switching centre.

The attraction of a star topology is that much of the intelligence required to control the network can reside in one place, that is, the central node. This is particularly suitable for simple terminal connection to devices which have little internal intelligence. It is quite commonly used for connecting all of the subscribers in a building, or on a campus, to a voice switch known as PABX (private automatic branch exchange) or a data switch. However, as the number of subscriber nodes grows, the central node becomes more complex. Furthermore, the topology is very vulnerable to failure of the central node, which results in failure of the whole network.

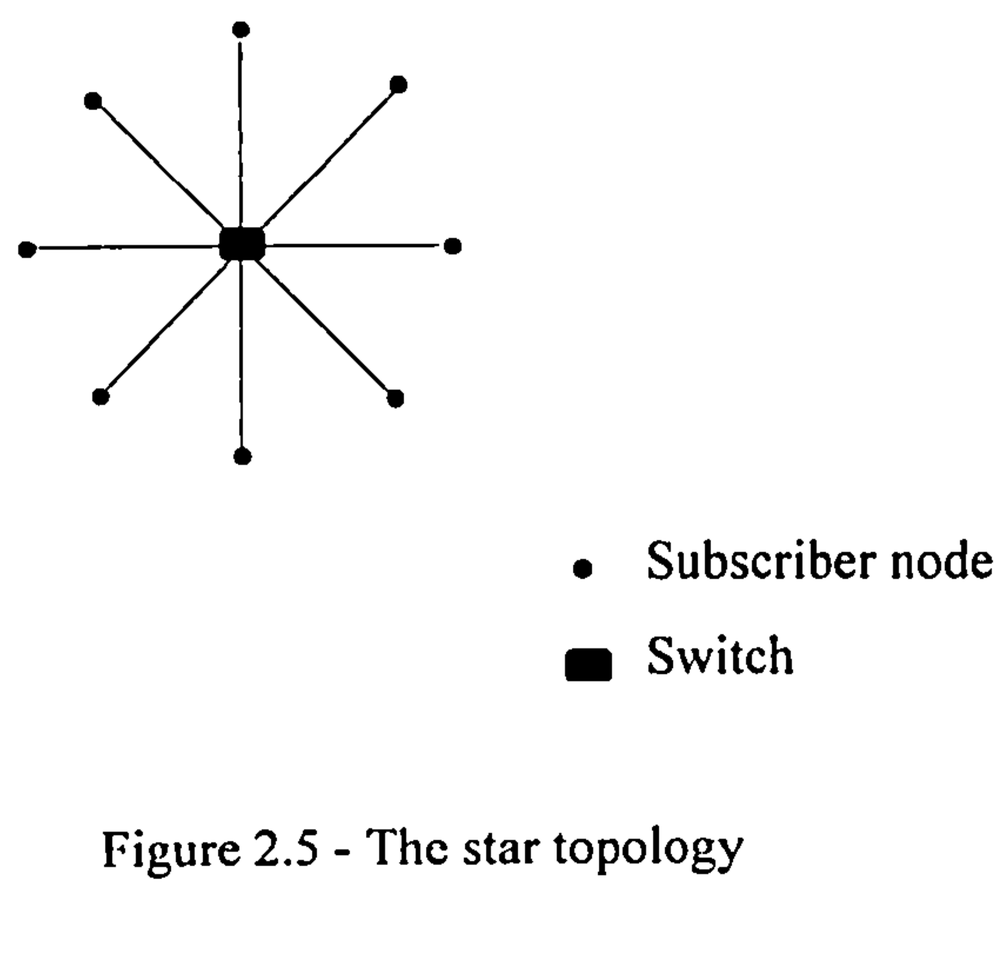


Figure 2.5 - The star topology

2.3.1.3 Bus topology

The bus topology employs a fully-shared broadcast transmission facility, called a bus, for exchanging information between nodes. Every station on the network is tied on to a common bus and can hear traffic from every other station. Simultaneous transmissions may occur, causing messages to be corrupted. Medium access protocols are required to avoid

these types of problems. Such a topology is generally the basis for a high-speed, distributed system used for interchange of information between computers and computer-controlled devices located within a building, or a campus. The bus topology is very popular in synthesising high-speed LANs that employ either the token or CSMA access methods. The Ethernet is based on a bus topology. Although logically the network is configured as using a bus, its physical layout becomes a star. It is therefore associated with the same unreliable features of a star connected network.

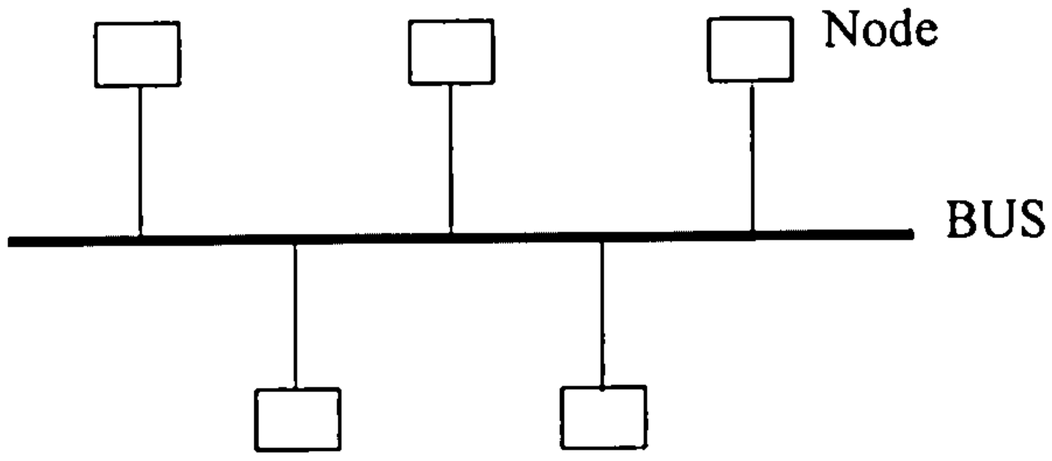


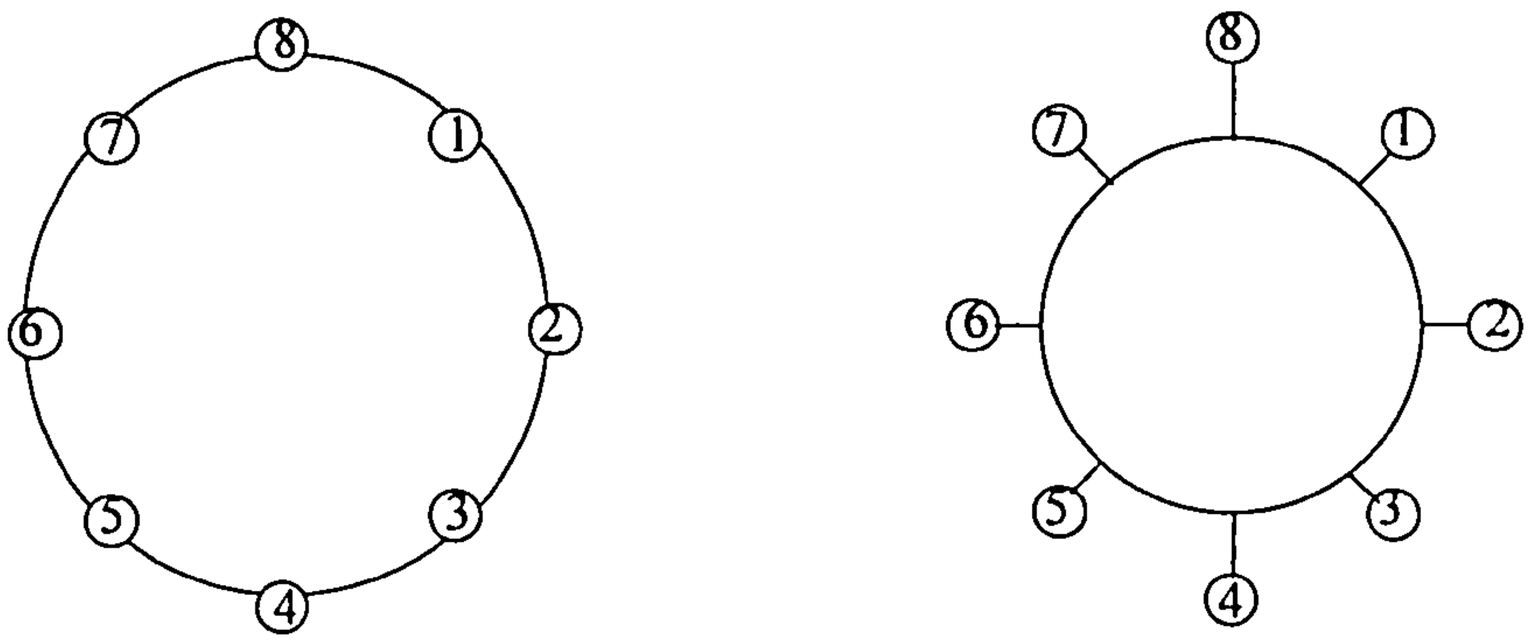
Figure 2.6 - The bus topology

2.3.1.4 Ring topology

A ring topology consists of a series of nodes connected to each other to form a closed loop. Figure 2.7(a) represents the store-and-forward type ring topology and Figure 2.7(b) represents a broadcast type ring system. In the store-and-forward type, each node is arranged to accept data transmitted to it from its immediate preceding neighbouring node to which it is connected, and then to forward the data to its immediate following node to which it is connected.

Many popular LAN architectures employ the ring topology using the token access. In these systems, a special control frame, called a token, is passed around the ring from one station to another, and only the station in possession of the ring token has the right to transmit. Transmission takes place in only one direction.

The ring topology suffers from the drawback that if the connection between any two nodes fails, the whole network is considered to have failed. Also, the token ring topology is physically configured as a star: since each station is connected to a wiring closet by its own line, with the result that failure of the wiring closet causes the whole network to stop operating. The Fibre Distributed Data Interface (FDDI) is a token ring, conceived as a backbone for Local Area Networks, which is to consist of dual counter-rotating rings such that if any failure occurs between two stations, the ring can be re-configured and hence remain operational.



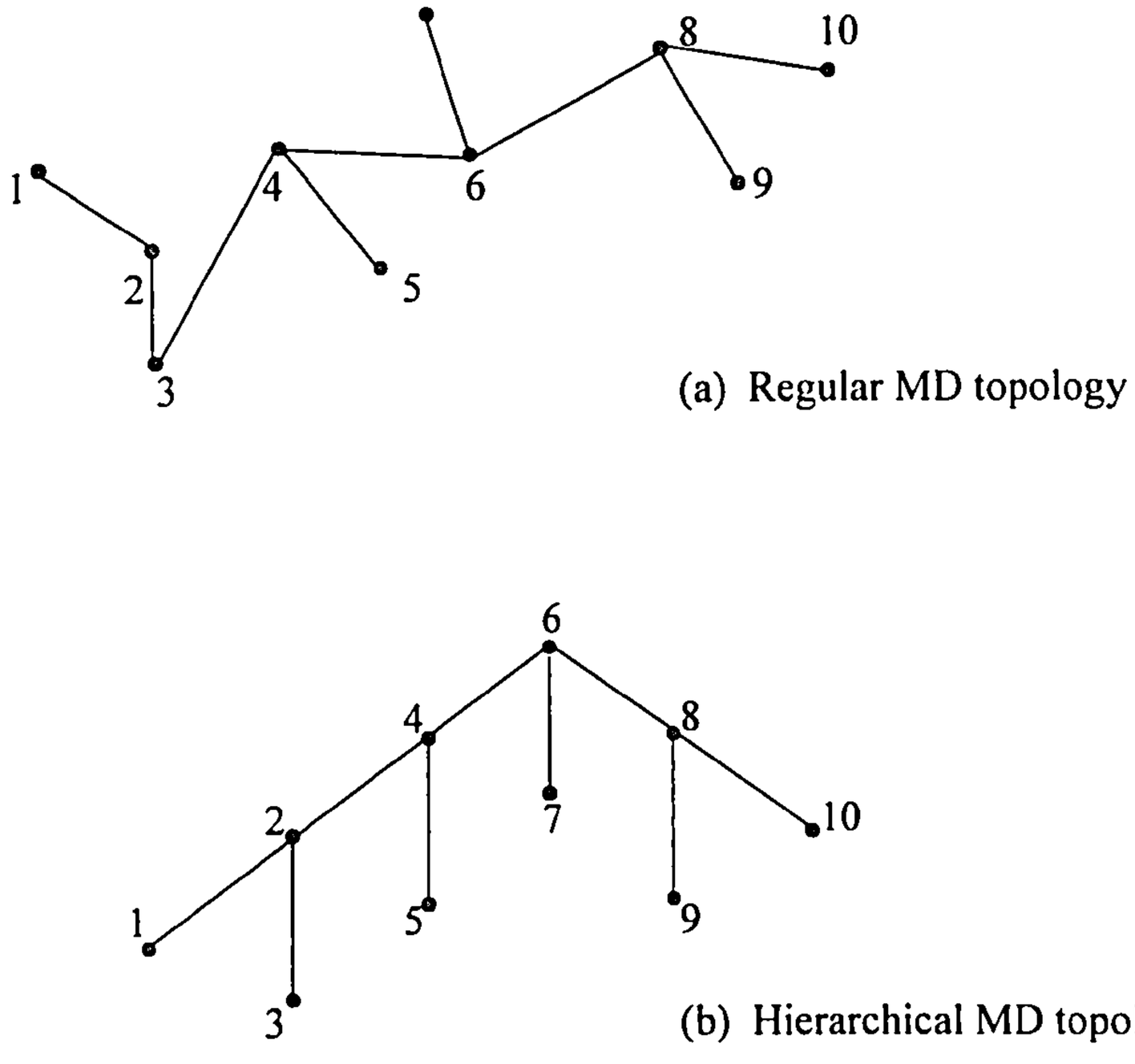
(a) S/F type ring topology (b) Broadcast type ring topology

Figure 2.7 - The ring topology

2.3.1.5 Tree or Multidrop (MD) topology

The tree-connected network, or multidrop topology, can occur in two forms, the general and the hierarchical topologies, as shown in Figure 2.8. The general MD topology finds its main use in the Local Area Network, whereas the hierarchical one is generally applicable to certain types of Wide Area Networks. Since there is only one path between any two nodes, routing is very simple. However, this also implies that if a link fails, it results in disconnection of the network.

A minimal spanning tree (MST) topology is a subset of the generalised tree or MD topology. This topology is by definition a network connecting all the nodes with no loops or closed paths, the sum of whose link costs is a minimum.



(a) Regular MD topology

(b) Hierarchical MD topology

Figure 2.8 - The multidrop topology

2.3.1.6 Mixed topologies

Many mixed topologies can be derived from the above mentioned basic topologies through the processes of superposition or interconnection. Since most of the cost of a communication network is dictated by the cost of communication links, network topology must be optimised for each application. Furthermore, the choice of topology depends on the level of hierarchy at which it needs to be applied.

The mesh topology is the only one which totally satisfies reliability requirements, if specified in terms of connectivity. In the star, bus and ring topologies, failure of the central facility, (the switching centre, the bus, or the wiring closet) results in major degradation whereas failure of any other component need not be so drastic. Failure of a link in the tree topology causes the network to be disconnected.

Given that one of the main aims of this thesis is to derive a method for designing fault-tolerant configurations, connectivity is an important aspect to consider, and the final network is expected to be in some mesh connected form, though it need not be fully connected.

2.3.2 Synthesis of network topologies [SHA90]

As mentioned before, the synthesis of a network is certainly not a one-time effort executed on a large mainframe computer at the beginning of a network planning cycle. There are no definite algorithms for optimising a network topology. The designer must employ a combination of heuristic algorithms and "what-if" analysis in an iterative manner until an acceptable solution is obtained. The solution consists of the following steps :

1. Given the traffic demands, compute the lower bound on the number of concentrators/ switches, as dictated by their maximum capacity to handle calls/messages/packets;
2. Based on the positions of the customer premise equipment, select the locations of the required number of concentrators/ switches;
3. Assign the customer premise equipment (CPE) locations to concentrators and the concentrators to switches;
4. Connect the CPE locations to concentrators to which they were assigned, and connect the concentrators to switches with the correct number of links, according to the desired topology;
5. Compute the total cost of the network derived for the allowable sets of design parameters and

6. Vary the set of design parameters to generate a number of different topologies. Compute the cost and evaluate the performance. Select the best solution.

In Chapter 4, the proposed design methodology assumes that steps 1 to 3 have already been executed and focuses on interconnecting the nodes, which could represent concentrators or switches. An iteration through steps 5 and 6 is used to generate a network which meets performability requirements.

Although steps 1 to 3 are not referred to in Chapter 4, for completeness, algorithms to execute these steps are described briefly in the following sections.

2.3.3 Computing the lower bound on the number of concentrators and switches [SHA90]

Computing the number of required concentrators and switches depends on two parameters:

1. the total number of calls / messages / bits expected to be handled during the busy hour and
2. the capacity of the concentrators and switches to handle traffic.

Based on the above variables, the designer can determine the minimum number of concentrators and / or switches required to design the network.

For voice applications, the CPE location is generally served by a PABX or a CO-Centrix. The first task is to calculate the optimum number of PABXs and/or CO-Centrexes, given the number of telephones at each customer location. Design is completed by determining the number and locations of tandem switches according to a least-cost criterion repeated for each selected number of locations (or PABXs) served.

For data applications, the number and type of data terminals at each location are generally given. The first task involves the computation of the minimum number of data concentrators needed by studying the capacity of a single concentrator in terms of messages or ports, or both. Next, the number of switches (starting from the minimum number of

switches required) for serving a given number of concentrators is varied until the least-cost criterion is met. This process may then be repeated by increasing the number of concentrators used, thereby generating different cost values. The designer can then choose an optimum network topology.

2.3.4 Algorithms for locating concentrators and switches

Having determined the number of concentrators and switches required, the designer then needs to place them in such a way as to achieve the least cost. The first sets of algorithms to be described tackle the problem of locating the concentrators, deciding how many to use out of a set, and which terminals to associate with each. The problem is applicable to a centralised network, where two levels are envisioned: terminals may be connected to a remote concentrator or linked directly to the central processor, and the remote concentrators in turn are then connected directly to the central facility.

The terminal locations are assumed known. *Possible* sites for the concentrators are also known; a subset of these concentrators is to be chosen, and the set of terminals associated with each is to be determined.

2.3.4.1 Problem formulation [SCH77]

The problem to be tackled can be described as follows: a number, n, of known terminals is to be connected, via concentrators, to a central facility, S_0. Call the set of terminals $\{T_1, T_2, \ldots, T_n\}$. A possible set of concentrator locations $\{S_1, S_2, \ldots, S_m\}$ is available, and a subset is to be chosen, with terminals assigned to each, or connected directly to S_0. (The set of concentrator locations can be chosen to be at terminal locations, if desired).

Concentrator S_j has a fixed cost f_j consisting of a fixed hardware cost and the communications cost of connecting S_0 to S_j. Each concentrator is assumed to have a maximum port, or terminal, capacity e. For the central facility S_0, let $f_0 = 0$ and $e_0 \geq n$. Let c_{ij} be the cost of connecting terminal i to concentrator j ($i = 1, 2, \ldots, n; j = 1, 2, \ldots, m$).

Two binary connection parameters are now defined.

$x_{ij} = 1$, iff T_i is connected to S_j

$= 0$, otherwise.

Call concentrator S_j is said to be *open* if it is in use and *closed* if it is not in use. Let

$y_j = 1$, if S_j is *open*

$= 0$, if S_j is *closed.*

Then, $y_j = 1$, iff $\sum_{i=1}^{n} x_{ij} > 0$

$= 0$, otherwise.

Therefore, the total cost to be minimised is

$$Z = \sum_{j=0}^{m} y_j f_j + \sum_{i=1}^{n} \sum_{j=0}^{m} x_{ij} c_{ij} \qquad (2.1)$$

The minimisation is to be carried out subject to the following constraints.

$$\sum_{j=0}^{m} x_{ij} = 1 \quad (i = 1,2,...,n) \qquad (2.2)$$

$$\sum_{i=1}^{n} x_{ij} \leq e \quad (j = 1,2,...,m) \qquad (2.3)$$

The problem then consists of finding the y_j's and x_{ij}'s that minimise Z subject to the two constraints. The algorithms to be described, the *add* and *drop* algorithms, are heuristic approaches for solving the defined problem.

2.3.4.2 Add algorithm [SCH77]

Initialisation consists of connecting all terminals to concentrator S_0. All concentrators are closed. The concentrators are then opened one at a time, in an iterative fashion, to provide a maximum decrease in cost. It is thus a steepest descent type of algorithm.

Steps of the algorithm:

1. Initialisation: connect all terminals to S_0. All other concentrators are closed. Evaluate cost of that configuration.

2. Each concentrator is opened, one at a time, to determine which one, if any, provides the greatest improvement over the initialisation step. For instance, open S_1 first and connect the terminals with the largest decrease in cost up to a maximum of e.

3. Repeat for each concentrator and open the one for which there is maximum gain in cost.

4. With this concentrator open, repeat step 2 to check if any further improvement is possible. Algorithm terminates when no further gain in cost can be achieved.

2.3.4.3 Drop algorithm [SCH77]

The algorithm starts with all concentrators open and then proceeds to close them one at a time, systematically and iteratively, until no further improvement is possible.

Steps of the algorithm:

1. Initialisation: terminals are connected to their nearest, that is least-cost, concentrator until the maximum number is accommodated by any concentrator.

2. Each concentrator is closed, one at a time, to determine which one provides the best improvement. For example, close S_1 and connect the terminals previously connected to it to other concentrators and then establish cost of the new configuration.

3. Repeat for each concentrator and close the one for which there is maximum gain in cost.

4. With this concentrator closed, repeat step 2 to check if any further improvement is possible. If not, the algorithm terminates.

For simple cases, *add* and *drop* algorithms can yield the same configuration, but in general they do not. Since the *drop* algorithm starts off by connecting the terminals to the nearest concentrators, the final solution may be arrived at using fewer iterations than the *add* algorithm.

2.3.4.4 Sharma's algorithm for locating concentrators and switches [SHA90]

Given the number of concentrators and switches required, Sharma has proposed a simple algorithm for locating them. He presumes that, in order to achieve least cost, the concentrator or switch has to be placed at an existing CPE location. This algorithm is based on the concepts of centre of gravity (COG) and successive decompositions, as explained in the following steps and Figures 2.9(a), (b) and (c).

1. Define the vertical (V) and horizontal (H) co-ordinates of each of the N CPE locations.

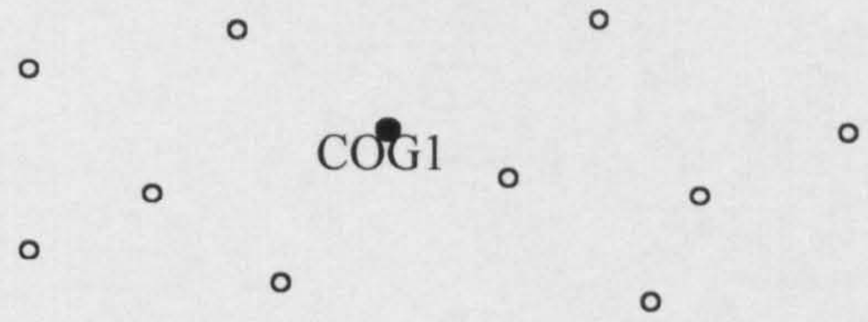
2. Find the V- and H- co-ordinates of the mathematical COG for all N locations, using the following expressions:

$$V(COG) = \left[\sum_{i=1}^{N} V_i.W_i\right] / \left[\sum_{i=1}^{N} W_i\right] \quad and \tag{2.4}$$

$$H(COG) = \left[\sum_{i=1}^{N} H_i.W_i\right] / \left[\sum_{i=1}^{N} W_i\right] \tag{2.5}$$

where V_i and H_i are the V(vertical) and H(horizontal) co-ordinates of the i^{th} locations, and W_i is the time-consistent average of the two-way traffic handled by the i^{th} location.

3. Find the closest CPE location to the above mathematical COG. This location is suggested as the optimum location for a single concentrator, or a switch.



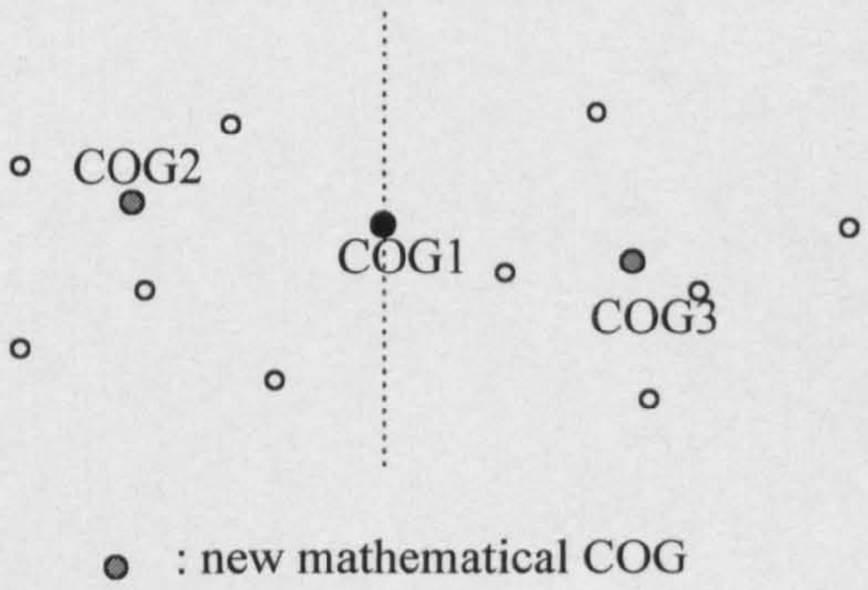
o : customer premise equipment location

● : mathematical centre of gravity

Figure 2.9(a) - Locating one centre of gravity

Successive decompositions are applied to determine the locations of more than one centre of gravity. Figure 2.9(a) is used as an example.

4. To find the locations of two COGs, the CPE locations are divided into two sets by a vertical line drawn through the mathematical COG. For each of these sets, a new mathematical COG is calculated using the procedure in step 2. This is shown in Figure 2.9(b). Assign the closest CPE locations to these two new COGs.



: new mathematical COG

Figure 2.9(b) - Locating 2 new centres of gravity

5. To find the locations of four COGs, two horizontal lines are drawn through the two COGs to divide the area covered in four regions, as shown in Figure 2.9(c). For each of the newly decomposed areas, step 2 can be applied to determine the centre of gravity. Assign the closest CPE locations to these four COGs.

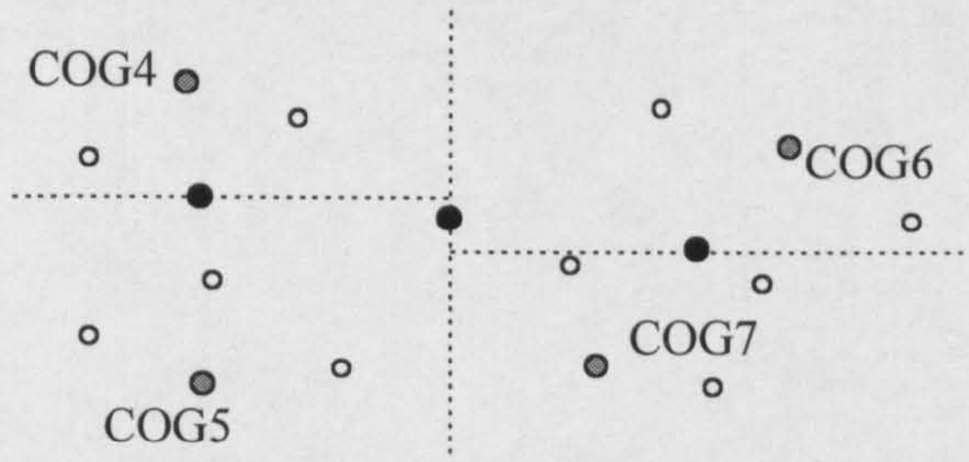


Figure 2.9(c) - Locating 4 new centres of gravity

6. Repeat Steps 4 and 5 by decomposing additional regions and computing the COG locations and the corresponding closest CPE locations. The alternation of vertical and horizontal decompositions is necessary to eliminate any dependence on geographical boundaries.

By naming these COGs and corresponding CPE locations in a consecutive manner, one can determine the entire solution space; and, 1, 2, ..., M locations can be selected for concentrators, or switches. Although the above algorithm provides a rather good solution in most cases, there are some pathological cases where the initial solution can be improved. One method is to study each of the M clusters and relocate the corresponding network centre/switch such that the sum of the costs for all access lines of that cluster is minimised.

$$Cost = \min_{i}\left[\sum_{j} d_{ij} w_{j}\right] \tag{2.6}$$

where

d_{ij} = distance or cost between the i^{th} centre and j^{th} location/node and

w_j = weight of j^{th} location/node in terms of millierlangs or the number of bits per second.

An algorithm based on an exhaustive search can yield an optimal solution but it will be very time consuming.

2.3.5 Algorithms for connecting CPE locations to concentrators and switches [SHA90]

The designer employs a bottom-up approach to connect all of the nodes to one another. At the lowest level, the CPE nodes need to be connected to the concentrators belonging to the next hierarchical level, according to an applicable topology. Then, these concentrators take on the role of unconnected nodes which must be linked to the concentrator nodes of the next level. This process is repeated until all nodes of a network hierarchy are connected together.

As the process of connecting the network evolves, a number of tasks need to be accomplished.

1. Accumulate the traffic intensity at each concentrator.
2. Accumulate the cost of network facilities, that is nodal hardware and transmission line, at each concentrator level of network hierarchy.

The above tasks are done in steps 5 to 8, 11, and 12 of the generalised design methodology given in Section 2.3.6.

2.3.6 A generalised algorithm for designing all network topologies [SHA90]

Assuming that a network of M nodes needs to be designed and traffic demands between the various nodes are known, a design algorithm must generally perform the following tasks:

1. Compute the number C of concentrators/switches required.
2. Find the optimum locations for C concentrators/switches.
3. Assign the unconnected nodes to the various concentrator/switch locations based on the closest distance resulting in C sets of locations. A more complex assignment criterion can be used.
4. Execute the steps of an algorithm required for designing any specific topology, for example a star topology, a mesh topology, or a minimal spanning tree.
5. Compute the total traffic (erlangs/bps/etc.) handled by each concentrator/switch by adding the traffic between each nodal pair.

6. Compute the number of circuits necessary to handle the required traffic on each access line (AL) bundle, while obeying the design constraint of, for example, the allowable data rate.
7. Compute the cost of each AL bundle as a function of the distance, the number of circuits in the bundle and the applicable tariff.
8. Compute the cost of each region and the entire system.
9. Repeat Tasks 1 to 8 for each level of concentrator until only the backbone network remains to be designed.
10. Synthesise the backbone network and connect the *S* switches that constitute it.
11. Compute the size of each trunk bundle as a function of the traffic handled, while obeying the design constraint of, for example, the allowable data rate.
12. Compute the cost of each trunk bundle as a function of the distance, number of trunks in the bundle and the applicable tariff. Also compute the cost of the entire trunk network.
13. Repeat tasks 1 to 12 for each combination of *C* concentrators and *S* switches and for various combinations of transmission and nodal hardware technologies.
14. Plot a graph of network related costs as a function of the number of switches. Choose the most optimum network topology consisting of the correct number of concentrators and switches for a correct combination of nodal/transmission technologies.
15. Display all of the resulting network topologies in the form of graphics with a summary of network costs.

2.3.7 Design of a specific topology

Task 4 is the one that needs to be changed in order to design the network according to a specific topology. Whereas design of some topologies is straight forward, for example the star network, a number of specific algorithms can be identified for design of others, for example the minimal spanning tree. Designing a star network only involves connecting each CPE to the associated concentrator, or switch, by an access link bundle.

In the design methodology to be presented in Chapter 4, the algorithm starts by generating the minimal spanning tree that connects all the nodes. A minimal spanning tree

guarantees that a minimal sum of lengths, or costs, is associated with all links employed to connect the nodes. The different existing algorithms to design the multidrop topology with or without constraints are reviewed in Section 2.3.8. Derivation of a mesh topology is treated as a case of distributed network design in Section 2.3.9.

2.3.8 Algorithm for designing a multidrop (MD) topology

Prim [PRI57] and Kruskal [KRU56] were the first to develop algorithms for connecting a set of nodes to one another according to a minimal spanning tree topology (MST). The original algorithms were derived with no regard to capacity constraints. Esau and Williams [ESA66] devised the first practical algorithm for designing an MST with capacity constraints. Kershenbaum and Chou [KER74] have presented a unified, parametrised algorithm for designing a multidrop topology with capacity constraints, and Chandy and Russell [CHA72] have used a branch and bound technique which provides a minimum cost solution to a constrained network design problem where the constraints imposed could relate to time, reliability or flow across the links.

2.3.8.1 Heuristic algorithms

The application of Prim's algorithm and Kruskal's algorithm for generating the minimal spanning tree when there are no capacity constraints, are as described in Sections 2.3.8.1.1 and 2.3.8.1.3. As expected, they provide the same solution to any problem. In the more practical case where capacity constraints are imposed, three heuristic algorithms are of interest: Prim's algorithm and Kruskal's algorithm are modified to cope with constraints, and the third one is due to Esau and Williams [ESA66]. Heuristic algorithms generally provide sub-optimum solutions that are quite close to the optimum but at a considerable saving in computation time. Upon the removal of constraints, all three algorithms produce a minimal spanning tree solution, but differ in their running time requirements.

These three algorithms will be described as solution methods for a centralised multipoint problem, which consists of finding the least costly network to connect a number of nodes to a central facility under the constraint that the capacity of any link should not exceed a specified value. The Esau-Williams algorithm essentially searches out the nodes

that are furthest from the centre in a cost sense, and connects them to neighbouring nodes that provide the greatest cost benefit. Prim's algorithm does the reverse: initially it selects the node closest to the centre, then connects in those nodes that are closest to those already in the network. Kruskal's algorithm simply connects the least-cost links, one at a time.

In the algorithms to be described, a cost matrix is identified where an entry c_{ij} corresponds to the cost of establishing a link between nodes i and j. The central facility is referred to as node 1.

2.3.8.1.1 Prim's algorithm for designing an MST topology without capacity constraints [SHA90]

The major steps for designing an MST network with no capacity constraints, using Prim's algorithm are described below.

Steps of the algorithm:

1. Start out with all unconnected, or isolated, nodes and no clusters.

2. Consider the closest node to an unconnected node. If the closest node is also unconnected, make a connection between the two nodes. This step, when repeated for all isolated nodes, will yield either isolated nodes or clusters of connected nodes.

3. Connect every isolated node to the closest cluster by the shortest connection.

4. Connect the two closest clusters by the shortest connection.

5. Repeat the above steps until only a single cluster is formed.

2.3.8.1.2 Prim's algorithm for designing an MD topology with capacity constraints [SCH77]

Assume that the centre node or concentrator is referred to as node 1. Each node is initially assigned a weighting factor w_i, with $w_1 = 0$; $w_i = -\infty$, $i \neq 1$. A trade-off function is defined as

$$t_{ij} \leftarrow c_{ij} - w_i$$

Initially, all $t_{ij} = \infty$, except t_{1j}, representing the cost of connecting each node to the centre. The only node initially connected in the network is thus the central node 1.

Steps of the algorithm:

1. Find the minimum t_{ij}. The algorithm thus searches out the minimum cost of connecting a node to one already connected in the network.

2. Check to see if the constraints are violated. If not, go to step 3. If yes, set the min $t_{ij} = \infty$, and return to step 1.

3. Add link *i-j*, setting $w_j \leftarrow 0$, readjust the constraints and recalculate all t_{ij}. Then go back to step 1.

4. Algorithm terminates when all the nodes have been connected.

2.3.8.1.3 Kruskal's algorithm for designing an MD topology without capacity constraints

The steps of Kruskal's algorithm are similar to Prim's algorithm except that the connections are made starting with the lowest cost link.

Steps of the algorithm:

1. From the table of costs of connecting two nodes *i* and *j*, select the least cost link.

BIBL. LONDIN. UNIV.

2. Add link. If all nodes are not connected, go back to step 1 to find next link in order of increasing cost.

3. Algorithm terminates when all the nodes have been connected.

2.3.8.1.4 Kruskal's algorithm for designing an MD topology with capacity constraints [SCH77]

Only Step 2 needs to be changed to verify if constraints are not being violated. The steps are as described below.

Steps of the algorithm:

1. From the table of costs of connecting two nodes *i* and *j*, select the least cost link.

2. Check if incorporation of that link violates any constraints. If yes, ignore link and go back to step 1 to find next link in order of increasing cost. If no, connect the two nodes and if all nodes are not yet connected, go back to step 1 to select the next link.

3. Algorithm terminates when all the nodes have been connected.

2.3.8.1.5 Esau-Williams algorithm for designing MD topologies with capacity constraints [SCH77]

There are a number of variations of the Esau-Williams algorithm and the description below is as given by Schwartz [SCH77].

Steps of the algorithm:

1. Initialisation: calculate all trade-off parameters t_{ij} for all nodes,

   where $t_{ij} = c_{ij} - c_{i1}$

   c_{ij} is taken from the cost matrix; the parameter t_{ij} measures the difference in cost between connecting node *i* to node *j*, and node *i* directly to the centre.

2. Select the minimum t_{ij} and consider connecting i to j.

3. If constraints are satisfied, go to Step 4. If not, set $t_{ij} = \infty$, go back to Step 2 and select again.

4. Add link *i-j*; label node i with the label of node j ($i \leftarrow j$) to show i is connected to node j; re-evaluate the constraints and update the trade-off parameters. Go back to Step 2.

5. Algorithm terminates when all links have been connected.

2.3.8.2 Kershenbaum and Chou's algorithm [KER74] [SCH77]

Kershenbaum and Chou have shown that the three heuristic algorithms described above, and others quoted in the literature, can be considered as special cases of a unified algorithm appropriate for solving the multipoint connection problem. The algorithm is essentially a modified version of Kruskal's algorithm where, instead of successively connecting in more and more costly links, the cost less a weighting factor is used as a measure of comparison.

Let c_{ij} be the cost of link *i-j*, connecting terminals i and j. Let w_i be the weight associated with terminal i. Then the trade off function is defined as

$$t_{ij} = c_{ij} - w_i$$

The values of w_i change during the execution of the algorithm and different definitions of w_i produce the various algorithms already discussed. The table below indicates the initial values of and the changes to the weighting factor appropriate to each algorithm.

Algorithm	Initialisation	Update when *i-j* is brought in
Kruskal	$w_i = 0,\ i = 1,\cdots,N$	none
Prim	$w_1 = 0,$	
	$w_i = -\infty,\ i = 2,\cdots,N$	$w_j \leftarrow 0$
Esau-Williams	$w_i = c_{i1},\ i = 1,\cdots,N$	$w_i \leftarrow w_j$

Table 2.1 - Table of weights

Steps of the algorithm:

1. Initialise the w_i for $i = 1, 2, \ldots, N$. Initialise the constraints. Set $t_{ij} \leftarrow c_{ij} - w_i$ for all i, j when c_{ij} exists and no constraints are violated with the connection i - j.

2. Find $t_{i'j'} = \min_{\substack{i,j \\ i \neq j}} t_{ij}$.
   If $t_{i'j'} = \infty$, terminate algorithm. Otherwise go to step 3.

3. Evaluate the constraints under the connection i' - j'. If they are violated, set $t_{i'j'} = \infty$, go to step 2. If not, go to step 4.

4. Add link i' - j'. Re-label one of the nodes to correspond to the other. Connection is made. Re-evaluate constraints. Update w_i as in table, re-evaluate t_{ij}, go to step 2.

2.3.8.3 Chandy and Russell's algorithm [CHA72] [SCH77]

This algorithm is believed to provide the optimum design for the constrained multipoint network. However, it might require very large running times especially for large networks. The approach, based on the branch and bound technique, is to partition the complete set of possible connections (or feasible solutions) into smaller and smaller subsets, always testing to see if a lower bound based on a minimum spanning tree connection for that subset is reached. When the lower bound is reached, the algorithm terminates.

Steps of the algorithm:

1. Given n nodes (terminals, concentrators and switches) and an n x n symmetric cost matrix whose $i-j^{th}$ element is the cost of connecting points i and j, the minimum spanning tree which connects all of them is determined.

If the solution to the unconstrained minimum spanning tree is feasible, then it is also optimal and the algorithm terminates. Otherwise, the cost associated with the MST is used as a lower bound on the cost of a feasible solution to the constrained problem.

2. The algorithm proceeds by partitioning the set of all feasible solutions, U, into smaller and smaller subsets. Each step produces a partition and the partition gets finer at each succeeding step. Lower bounds are determined.

(a) First partition : Divide the set U into 2 subsets such that one subset A includes all the direct links between the nodes and the centre node or concentrator from the MST. These links are said to be *established* in A. Subset B is the complement of A, that is at least one of the links in A is not connected, that is *not allowed*. A link which is neither *established* nor *not allowed* is said to be *free*.

(b) Chandy and Russell claim that if the nodes $i_1,\ldots, i_r$ are directly connected to the centre in the unconstrained minimum spanning tree, then there exists an optimal solution to the constrained problem in which terminals $i_1,\ldots, i_r$ are directly connected to the centre.

(c) Therefore, the optimal solution to the constrained problem is bound to be in subset A and subset B can be disregarded. If a feasible solution appears in A, the algorithm stops; otherwise partitioning proceeds in the following way.

3(a) Let subsets in current i^{th} partition be written as $s_1^{(i)}, s_2^{(i)}, \cdots, s_k^{(i)}$ with k subsets in all. Let lower bounds on the costs be $L_1^{(i)}, L_2^{(i)}, \cdots, L_k^{(i)}$.

(b) Let $L^{(i)}$ be the lowest of the lower bounds. Say this is subset j so that $L_j^{(i)} = L^{(i)}$. Test $s_j^{(i)}$ for its feasible solutions. If a feasible solution exists that has cost $L_j^{(i)}$, it is the optimum solution and the algorithm terminates.

(c) If no feasible solution has cost $L_1^{(i)}$, re-label set j as set k. Then partition set $s_k^{(i)}$ into sets $s_k^{(i+1)}$ and $s_{k+1}^{(i+1)}$. The partitioning consists of connecting a previously *free* link in $s_k^{(i)}$ to form subset $s_k^{(i+1)}$ while not allowing this connection in $s_{k+1}^{(i+1)}$. The lower bounds $L_k^{(i+1)}$ and $L_{k+1}^{(i+1)}$ are computed and the procedure is repeated.

(d) For each partition, the cost $L_x^{(i)}$ consists of a fixed cost $U_x^{(i)}$ and a cost $Z_x^{(i)}$ where $U_x^{(i)}$ is the cost of connecting the established links for that partition and $Z_x^{(i)}$ is the cost of the minimum spanning tree involving the remaining links found. The lower bound increases or remains the same on successive partitions, that is $L^{(i+1)} \geq L^{(i)}$.

2.3.9 Distributed Network Design

Given the locations of the concentrators, distributed network design involves choosing the set of links to interconnect them, subject to cost constraints. Performance and reliability requirements also need to be met. Reliability is often specified in terms of nodal connectivity whereas performance can be expressed in terms of, for example, average network delay.

2.3.9.1 Cut Saturation algorithm [GER74] [SCH77]

The algorithm to be described is known as the *cut-saturation* algorithm. It iteratively attempts to keep the network throughput within specified bounds while reducing the overall line cost and maintaining capacity, delay and reliability constraints. For the sake of simplicity, link capacity is assumed to have been determined and is taken to be the same for all links. A more complex capacity assignment method may be incorporated in the algorithm. Traffic requirements between all pairs of nodes are also assumed to be identical. The time delay is minimised by a routing procedure that finds the optimum routing of

messages between all source-destination pairs after each network design iteration. Reliability is ensured by a 2-connectivity constraint so that there are at least two distinct paths between any two nodes.

The cut saturation algorithm consists of five basic steps in any one iteration:

1. Routing: the optimal links are found, for a given network design, that minimise the overall network time delay. The routing algorithm employed is based on an adaptation of the Flow Deviation Method for solving non-linear, constrained, multi-commodity flow problems.

2. Saturated cutset determination: once the optimal flows are found, the links are ordered according to their utilisation. The links are then removed, one at a time, in order of utilisation. The minimal set that disconnects the network is called a *saturated cutset*.

3. Add-only step: this adds the least-cost links to the network that will divert traffic from the saturated cutset. To divert traffic effectively, nodes on either side of and relatively far removed from the cutset should be linked. But the further apart the nodes are, the higher the communication cost. As a compromise, a *distance 2* criterion is used in which nodes that are at least two links from nodes adjacent to the cutset nodes are chosen as candidates for possible linkage.

4. Delete-only step: this step in the algorithm eliminates links from a highly connected topology. One link at a time is removed at each iteration, the one chosen being the link which is the most expensive and underutilised. The link which maximises the following criterion is removed.

$$\text{Maximise } E_i = D_i \text{ x } \frac{C_i - f_i}{C_i} \tag{2.7}$$

where

D_i is the cost of link i,

C_i is its capacity and

f_i is its flow.

5. Perturbation step: once a desired throughput level has been attained, the network links are rearranged, using *add-only* and *delete-only* operations, to reduce the cost. Upper and lower bounds of about ± 5% about the desired throughput are set, and *add-only* and *delete-only* operations are used sequentially so long as the throughput remains within the bounds. If the throughput $R < R_{min}$, where R_{min} is the lower bound, an *add-only* step is required to bring the throughput up. If $R > R_{max}$, where R_{max} is the upper bound, a *delete-only* step is used to reduce the throughput.

An additional step in the *cut saturation* algorithm involves collapsing chains. Chains are portions of a network with a sequence of serial nodes. A network satisfying the 2-connected criterion should contain many chains. Those chains carry predominantly transit traffic and can be *collapsed* or replaced by a single link to improve the efficiency of the algorithm.

2.4 Capacity and flow assignment

Optimum selection of capacities for access lines and trunks in a communication network is a problem commonly faced by network designers. Assuming that the topology and the traffic flows are given and fixed, an optimal network can be designed in the sense that the capacity assignment perfectly matches the network traffic flows under conditions of fixed routing, and delay constraints. Similarly, given the capacities and topology of the network, the traffic flows can be determined so as to maximise its performance. However, the joint capacity and flow assignment optimisation problem, which plays a central role in the design of large scale packet switched networks, is a more complicated problem. Although individual optimum solutions can be obtained for the capacity assignment and flow assignment problems, combination of the two does not necessarily generate a globally optimal solution.

2.4.1 The capacity assignment problem [BER87]

The capacity assignment problem consists of choosing the capacity C_{ij} of each link (i, j) so as to minimise the linear cost

$$\sum_{(i,j)} p_{ij} C_{ij}$$

where p_{ij} is a known positive price per unit capacity, subject to the constraints that the average delay per packet should not exceed a given constant T.

The flow on each link (i, j) is known and is denoted by F_{ij} and is expressed in the same units as capacity. Treating each link as an M/M/1 model, the average delay constraint is given by the following expression.

$$\frac{1}{\gamma} \sum_{(i,j)} \frac{F_{ij}}{C_{ij} - F_{ij}} \leq T \tag{2.8}$$

where γ is the total arrival rate into the network, that is, the sum of the input flow for each origin-destination pair. It is assumed that the routing mechanism is known and the input flows, which depend on routing, are known as well. The problem now reduces to minimising the linear cost over the capacities C_{ij} subject to the delay constraint. Intuitively, the constraint is satisfied as an equality at the optimum.

On introducing a Lagrange multiplier, β, the following Lagrangian function is formed.

$$L = \sum_{(i,j)} \left(p_{ij} C_{ij} + \frac{\beta}{\gamma} \frac{F_{ij}}{C_{ij} - F_{ij}} \right) \tag{2.9}$$

In accordance with the Lagrange multiplier technique, the partial derivatives $\partial L / \partial C_{ij}$ are set to zero, that is,

$$\frac{\partial L}{\partial C_{ij}} = p_{ij} - \frac{b F_{ij}}{\gamma \left(C_{ij} - F_{ij} \right)^2} = 0 \tag{2.10}$$

Solving for C_{ij} yields

$$C_{ij} = F_{ij} + \sqrt{\frac{\beta F_{ij}}{\gamma p_{ij}}} \tag{2.11}$$

Substituting into the constraint equation,

$$T = \frac{1}{\gamma}\sum_{(i,j)}\frac{F_{ij}}{C_{ij}-F_{ij}} = \sum_{(i,j)}\sqrt{\frac{p_{ij}F_{ij}}{\beta\gamma}}$$

$$\Rightarrow \qquad \sqrt{\beta} = \frac{1}{T}\sum_{(i,j)}\sqrt{\frac{p_{ij}F_{ij}}{\gamma}} \tag{2.12}$$

On substituting into equation (2.11), the optimal solution is obtained and found to be

$$C_{ij} = F_{ij} + \frac{1}{T}\sqrt{\frac{F_{ij}}{\gamma p_{ij}}}\sum_{(m,n)}\sqrt{\frac{p_{mn}F_{mn}}{\gamma}} \tag{2.13}$$

$$= F_{ij}\left(1 + \frac{1}{\gamma T}\frac{\sum_{(m,n)}\sqrt{p_{mn}F_{mn}}}{\sqrt{p_{ij}F_{ij}}}\right)$$

The above assignment mechanism allocates capacity such that each channel receives at least F_{ij}, which is its minimum required amount, and then allocates some additional capacity to each. This optimal capacity assignment method is referred to as the *square root channel capacity assignment* method [KLE76].

2.4.2 The flow assignment problem [KLE76]

The flow assignment problem is the converse of the capacity assignment problem, where capacities are given and flows must be determined so as to minimise the average delay. The method to be described, the *flow deviation method*, is due to Fratta, Gerla and Kleinrock [FRA73] and gives an exact solution to the flow assignment optimisation problem which is computationally efficient. The steps of the algorithm are outlined below.

Let $\mathbf{f}^{(n)} = (F_1^{(n)}, F_2^{(n)}, \cdots, F_M^{(n)})$

where $\mathbf{f}^{(n)}$ is the flow vector at the n^{th} iteration of the algorithm whose i^{th} component $F_i^{(n)}$ represents the total flow on the i^{th} channel at the n^{th} iteration. It is assumed that there is a feasible starting flow $\mathbf{f}^{(0)}$; and the procedure to determine this flow is not described here.

Flow deviation optimal routing algorithm:

1. Let $n = 0$

2. For each $i = 1, 2, \ldots, M$ compute

$$l_i = \frac{C_i}{\gamma\left[C_i - F_i^{(n)}\right]^2} \tag{2.14}$$

where l_i is a "length" associated with the i^{th} branch and is given by $\partial T / \partial F_i$ for

$$T = \frac{1}{\gamma}\sum_{i=1}^{M}\frac{F_i}{(C_i - F_i)}. \tag{2.15}$$

3. Compute β_n, the incremental cost rate at this flow where

$$\beta_n = \sum_{i=1}^{M} l_i F_i^{(n)}. \tag{2.16}$$

4. Solve the shortest route problem using the lengths l_i. Let ϕ_i be the resultant flow on the i^{th} channel which would result if all flows were routed via these shortest paths. Let the vector of these flows be denoted by

$$\boldsymbol{\phi} = (\phi_1, \phi_2, \cdots, \phi_M).$$

5. Compute b_n, the incremental cost rate for the shortest route flow where

$$b_n = \sum_{i=1}^{M} l_i \phi_i. \tag{2.17}$$

6. Stopping rule: if $\beta_n - b_n < \varepsilon$ where $\varepsilon > 0$ is a properly chosen acceptance tolerance, then stop. Otherwise, go to step 7.

7. Find that value of α in the range $0 \le \alpha \le 1$ such that the flow $(1-\alpha)\mathbf{f}^{(n)} + \alpha\boldsymbol{\phi}$ minimises T. Let this optimum value be denoted by a. This optimum value may be located by any convenient search method such as Fibonacci search.

8. Flow deviation: let $\mathbf{f}^{(n+1)} = (1-a)\mathbf{f}^{(n)} + a\boldsymbol{\phi}$.

9. Let $n = n + 1$. Go to step 2.

The procedure to find a starting flow will either find a feasible starting flow or will declare the problem not solvable within the tolerances chosen. The *flow deviation* algorithm turns out to be an optimum procedure for solving the flow assignment problem.

2.4.3 The capacity and flow assignment problem [KLE76]

Kleinrock [KLE76] presents the capacity and flow assignment (CFA) problem in its dual form in the following way.

Given : Network topology

Minimise : $D = \sum_i d_i(C_i)$ where D is cost of entire network (assumed to consist only of channel constructions)

With respect to: $\{C_i\}$ and $\{\lambda_i\}$ which are capacity and flow across link i respectively.

Under constraints: $T = \sum_i \frac{\lambda_i}{\gamma} T_i \leq T_{MAX}$ where T is the average network delay.

The approach to finding a local minimum is to begin with a feasible starting flow, calculate the optimum *capacity assignment* under linearised costs, carry out a *flow deviation* algorithm [KLE76] to find the optimum flows, repeat the *capacity assignment* [KLE76] problem for these new flows and continue to iterate between the *capacity assignment* solution and the *flow assignment* solution until a local minimum is found.

Under the assumption of a feasible starting flow $\mathbf{f}^{(0)}$, the steps of the algorithm are as given below.

1. Let $n = 0$.

2. Carry out the *capacity assignment* algorithm for the flow $\mathbf{f}^{(n)}$ to find an optimal set of capacities, using linearised costs.

3. Using the lengths $l_i = \partial T / \partial F_i$, carry out the *flow deviation* algorithm in which $\alpha = 1$ at each step. Let the optimum flow so obtained be denoted by $\mathbf{f}^{(n+1)}$.

4. If T for the flow $\mathbf{f}^{(n+1)}$ is greater than or equal to T for the flow $\mathbf{f}^{(n)}$, then stop; the flow $\mathbf{f}^{(n)}$ is the local minimum obtained. Otherwise, let $n = n + 1$ and go to step 2.

The algorithm can be repeated for a number of different feasible starting flows in an attempt to find a global minimum.

In the discussion of capacity assignment, flow assignment, and the joint capacity and flow assignment problems, the underlying assumption has been that the network is and remains in a particular configuration. Therefore, if capacity is assigned to the links to meet demands and performance requirements in a fully operational configuration, the network is not necessarily equipped to face a state of failure. Performance may degrade drastically in the presence of a single component failure. There is no consideration for the aspect of fault-tolerant design. In the next section, fault-tolerant design based on capacity augmentation is discussed.

2.5 Fault-tolerant design - capacity augmentation

Given a network topology where channel capacities and flows have been determined, capacity augmentation can be used as a design technique for packet switched networks that prevents the performance degradation caused by network failures. This is achieved by increasing the capacity of the links sufficiently so that they can cope with any excess traffic due to re-routing upon the occurrence of failures. The proofing method [NIS88] [YOK91] is a method which makes use of this technique.

2.5.1 Proofing method [NIS88]

The proofing method determines capacity assignment for each link to meet the following design criterion. The network delay does not exceed design parameter T_d, as long as there is at least one path between any pair of nodes, even if the network fails. In case of failure, the method should provide alternative routes which are geographically separated. It is based on the technique wherein capacity for a link is augmented in advance, so that the congestion caused by detoured traffic can be avoided. A detailed description of the method is given below.

1. Generate the m most probable fail states by using the ORDER algorithm [LI84]. Value m is a number large enough to cover a high proportion of the total state space.

2. Solve the capacity and flow assignment problem [KLE76] for each state so that the network cost is minimised under the constraint of T_d, and obtain the capacity for each link CAP(i, j) which means the capacity of link i in state S_j.

3. Determine the capacity for link i, C_i from the augmented capacity function defined as

$$C_i = \sum_{j=0}^{m-1} \alpha_{ij} P_j \, CAP(i,j) \tag{2.18}$$

where P_j is the probability associated with state S_j and α_{ij} is a weighting factor for link i in state S_j and represents its level of importance. The weighting factor α_{ij} is referred to as the bias value.

The crucial point in the proofing method is how to determine the value of m and the bias value. The value of m is set, based on how robust the network must be. It determines the number of failure states that are considered so that the higher the value of m, the more robust the network is expected to be. Network cost and performance are closely dependent on the bias values. It is difficult to estimate the bias values which satisfy a given design criterion. The following two heuristic approaches have been adopted [NIS88].

2.5.1.1 Max delay method [NIS88]

The maximum capacity among those calculated in all states is assigned to all links. Let S_{k_i} denote the state where the maximum capacity is assigned to link i ($1 \le i \le M$) among all states considered. The bias value is defined as

$$\alpha_{ij} = \begin{cases} 1/P_j & \text{if } j = k_i \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

for such k_i as $\mathrm{CAP}(i, k_i) \ge \mathrm{CAP}(i, j)$, $1 \le i \le M, j = 0, \ldots, m-1$.

This method is the most reliable in the sense that the above criterion is always satisfied. It is, however, the most expensive approach.

2.5.1.2 Max average method [NIS88]

This method differentiates between the links to which capacity is not assigned and those to which a finite capacity is assigned in state S_0. The links to which capacity is not assigned in state S_0 are referred to as zero-capacity links. Each zero-capacity link is assigned the maximum capacity out of the values calculated for this link in all the other states. For other links, a uniform importance level is assumed to be taken. Let n and S_{k_n} denote a zero-capacity link and the state in which the capacity of link n is the maximum respectively. The bias value becomes

$$\alpha_{ij} = \begin{cases} 1/P_j & \text{if } i = n, j = k_n \\ 0 & \text{if } i = n, j \ne k_n \\ \alpha & \text{otherwise} \end{cases} \tag{2.20}$$

for all such n and k_n as $\mathrm{CAP}(n, 0) = 0$ and $\mathrm{CAP}(n, k_n) \ge \mathrm{CAP}(n, j), j = 0, \ldots, m-1$. α is a constant value. This method treats the zero-capacity links especially based on the following fact. When a uniform bias value is chosen for all links, sufficient capacity for accommodating the detoured traffic is not assigned to the zero-capacity links, because the

state probability of S_0 is so large that the terms of other states have little effect on the summation of equation (2.18).

The max-delay and max average methods cannot prevent a state where there are no routes between a particular source and destination node pair. To cope with this, a topological design method must be considered as well. In both cases, the performance metric used, which is a performance-related reliability measure, is the weighted sum of the performance in each network state; the weight is the probability of a state occurrence.

The proofing method involves a steady-state analysis of the network being considered. The probability associated with each state corresponds to its steady-state probability. The system's evolution is not observed over a period of time. The method also excludes the notion of repair. The proofing method requires the solution of the joint capacity and flow assignment problem in each state considered, which can be a fairly intensive and expensive procedure.

2.5.2 Fault-tolerant optimal channel allocation and flow assignment problem [KAN92]

Kang and Tan [KAN92] have proposed a design method which considers the joint capacity and flow assignment optimisation problem in environments subject to topological state changes that result from link failures. The aim is to obtain fault-tolerant schemes in which traffic demand and QOS network performance requirements are met, regardless of the network topological states.

A finite set of network states is used to represent different link failure configurations. The optimal capacity allocation / flow assignment (OCA/FA) problem is formulated as follows: suppose the traffic demand requirement Γ is given along with a QOS requirement specified in terms of a maximum allowable performance threshold P_{max} imposed on a performance measure in each state. The objective is to simultaneously determine the channel configuration and the flow configuration in each state, s, to minimise the total network cost with respect to the performance constraint $P_s \leq P_{max}$. Average delay

and buffer overflow probability QOS performance measures are considered. The network cost is assumed to consist only of link costs and the cost of each link is a monotonically increasing convex function of the number of channels over that link.

The above problem can be separated into two sub-problems where the first sub problem determines the optimum channel allocation for given flow assignments in every state; and the second involves the determination of the optimum QOS flow assignments in each state. The concept of marginal analysis is applied to obtain an efficient incremental allocation solution to sub-problem 1. For each state, sub-problem 2 can be tackled using the *flow deviation* algorithm or any other suitable optimal routing algorithm. These two sub-problems can be iterated sequentially until there is no further improvement in reducing the network cost. In solution of sub-problem 1, the approach is to choose the state of worst performance and augment the capacity of the link, which causes most improvement in performance in that state. No consideration is given to how often the network is in that state and for how long, which depend on the failure rate and mean repair time of the component that fails respectively.

2.6 Conclusions

In this chapter, a discussion of the overall design problem has been presented. Design of communication networks consists of the complex task of finding the right combination of parameters, such as topology and capacity, to meet requirements in terms of performance and reliability, within the limits imposed by a budget. The design problem cannot be handled in one execution of some algorithm. It needs to be approached in an iterative manner.

In the literature and in the thesis, the design problem is decomposed into three sub-problems: topological design, capacity assignment, and routing. Routing has been treated as a case of a flow assignment problem; and different routing algorithms are described in Chapter 4. In Chapter 2, emphasis has been placed on the sub-problems of topological design and capacity assignment.

Topological design, in itself, is complex due to the number of steps that are involved. A number of specific topologies have been used in practice in design, and a discussion of these topologies is given in this chapter. Then, the steps which constitute the synthesis of a network are outlined. Algorithms for each step are given for completeness, even though not all of them are used in the design methodology proposed in Chapter 4. In particular, the design of the minimal spanning tree has been presented as a case of a centralised design problem, whereas the design of the mesh topology was treated in distributed network design. In topological design, cost acts as a constraint and it has to be minimised with performance and reliability requirements being specified separately. Depending on the topology being designed, reliability can be expressed in terms of connectivity. For example, in the cut-saturation method, a 2-connectivity criterion is judged to be sufficient to meet reliability specifications, and performance is then optimised assuming failure-free conditions.

The sub-problems of capacity and flow assignment have also been discussed. Algorithms for the individual as well as the joint capacity and flow assignment problems have been given in the chapter. The main objectives are to optimise performance while minimising cost. Each of these algorithms is applicable to the network operating in a particular configuration and does not take into consideration possibilities of occurrence of component failures.

Nishida and Miyahara [NIS88], and Kang and Tan [KAN92] have proposed capacity augmentation methods as means of fault-tolerant design. Both methods have been described. The proofing method solves for the capacity and flow assignment problem in each state in which the network can be; assignment of capacity to a link is then based on the probability of being in each state and the capacity it was allocated in that state. Kang and Tan increase capacity iteratively; they select the state of worst performance, augment the capacity of the link which results in best improvement in performance in that particular state. They do not consider how each capacity augmentation affects performance in other states. Both methods are based on a steady-state analysis of the network, and do not view the evolution of the network over a period of time as it is affected by structural changes.

Although the probability of the network being in a state as a result of a failure is being considered, the duration of time spent in that state awaiting repair is not taken into account.

Therefore, firstly, traditional design methods treat performance and reliability separately. Both topological design and capacity assignment do not pay any attention to the frequency of failure of various components, nor to the time taken to bring them back to operation. Even if it ensured that the network remains connected when there is a component failure, there is no concern for how performance degrades upon the occurrence of that failure. In the capacity augmentation methods reviewed, the issue of repair is not taken into consideration.

In view of these shortcomings, a design method which combines the concepts of performance and reliability is desirable; a technique which takes into consideration performance in all states, rates of failures and mean repair times is necessary if a fault-tolerant approach to design is required. This calls for an investigation of the concept of performability, a combined measure of performance and reliability, in the design of communication networks.

Chapter 3

Performability : definition, model construction and solution

3.1 Introduction

With computing and communication systems becoming more and more complex, there is a growing need for modelling and evaluation techniques which are appropriate for them. Traditionally, as has been explained earlier, system evaluation has been done with respect to two different criteria: performance and reliability. These matters will now be considered in much greater detail in this chapter.

Performance refers to how efficiently, or effectively, a system delivers a specified service. Performance measures can be classified as being either resource-based or task-based. Typical resource-based measures include mean queue length at a server, throughput of a server or network, or utilisation of a server. Task-based measures refer to performance as perceived by the user and relate to figures such as response time, or some form of time delays. Several performance modelling tools exist, which are based on either analytical or simulation techniques. An analytical model is defined by abstracting away the less important details of a system. A computer or communication system can be represented by a product form queueing network, or some other mathematical model. Such models can be mathematically, or computationally, intractable for complex systems. In this case, approximation techniques are then needed. A simulation program is more appropriate if a detailed model of a complex system is required. In this approach, results are based on actual sample paths and are approximate so that they need to be given with confidence intervals. Simulation and analytical methods can be viewed as complementary techniques. The approach chosen depends on the system being modelled and the type of measures sought.

Generally and traditionally, performance modelling of a communication network is based mainly on queueing theory, and considers the system to be a network of queues and servers, characterised by arrival and service time distributions, size of buffers and queueing disciplines. The performance of the system is evaluated assuming that no structural changes

occur, that is, no links or nodes fail, or are repaired. The system remains in a particular configuration, or state, throughout the observation period. No consideration is given as to how the reliability of components affects the performance of the system during the specified interval.

System reliability, $R(t)$, is defined as the probability that a given system (component, or group of components) will operate without failure until a given time when it is used under specified environmental conditions [LIN80]. Typical figures which can be used to give an insight into reliability are failure rates, failure distribution functions, mean time between failures, or mean time to repair. Another important measure in reliability analysis is availability. Instantaneous, or pointwise, availability $A(t)$ is defined as the probability that a system is operational at time t, regardless of how often it has failed during the period of time [LIN80]. Interval availability, $A(0, T)$, is the fraction of time for which the system is operational during the interval $(0, T)$ [LIN80].

Meyer defines reliability as a measure which quantifies "the continuous delivery of proper service" and availability as a measure which quantifies "the alternation between proper and improper service" [MEY92]. Service is judged to be proper as long as it is delivered at or above a specified level. System failure is identified with a switch from proper to improper service. Dependability is then the "trustworthiness of a system with respect to the delivery of the specified service" [MEY92] and includes reliability and availability as special cases. However, in general, reliability has been used to broadly mean dependability and both terms will be used interchangeably in this thesis.

Reliability modelling deals with the representation of changes in the structure of the system, which are generally due to faults. When performing the reliability analysis of a complex system, it is almost impossible to treat the system in its entirety. A more logical approach is to decompose the system into a number of functional entities, which consist of sub-systems or components. A block diagram of system operation can be generated using these sub-divisions. Models are formulated to fit the logical structure. Series and parallel structures often occur. Overall system reliability is then computed in terms of the reliability of the sub-divisions.

Such a decomposition is not viable in cases where failures of system elements are dependent, as a result of environmental or design effects, the standby nature of the structure, repair, or replacement. Structural aspects of the system can no longer be separated. In these cases, analysis techniques need to deal with joint density and distribution functions, or Markov, and other state-transition models.

In reliability modelling, the system is considered to be binary-valued with respect to performance. Performance is either good, in which case the system is operational, or bad so that the system is judged to have failed. There is no regard for the different levels of performance, which can be attained during a period when the system is functioning properly. Reliability analysis gives no insight as to how performance actually varies during the whole observation period, especially in the interval when the system is considered to be operational. Conversely, performance evaluation completely ignores the possibilities of the occurrence of failures, with or without subsequent repair. Separate assessments of performance and reliability do not suffice in determining the overall quality of a delivered service, especially in the case of degradable systems. For such systems, failures can reduce the quality of a delivered service even though that service is still judged to be proper according to its specification. Fault-tolerant systems can be included in this category whereby a hardware or software failure causes a degradation in performance but not necessarily unavailability. In order to appraise degradable systems, there has to be a means of taking into account the frequency of failures, how fast repairs can be done, and how performance varies as a result of these structural changes and other factors which affect the system. A measure is sought, which can accommodate this property of performance degradation, as a result of some failure, which does not necessarily mean that the whole system is no longer functioning properly. The need to characterise such systems has been the *raison d'être* for the emergence of the concept of *performability*.

Performability is a mission-specific measure of system effectiveness that seeks to combine traditional reliability and performance measures of a system [PAT89]. A performability assessment would consider the system under investigation over a period of time, and try to evaluate the probability that the system operates within a certain range of performance levels, taking into account the occurrence of failures and repairs. In order to

determine performability, a model needs to be derived and solved. The rest of this chapter is devoted to formally defining performability, constructing a suitable model and solving it.

A mathematical definition of performability is given in Section 3.2 and the choice of performance variable is considered briefly in Section 3.3. Following this, performability evaluation is described in Section 3.4, by looking at the construction of a performability model and its solution. The idea of the Markov Reward Model, used as means of modelling is introduced in Section 3.5 and Section 3.6 consists of a review of existing techniques to solve for performability. In this thesis, the *randomisation* technique is chosen as the solution method and it is described in detail in Sections 3.7 and 3.8. The application of the *randomisation* technique to a communication networks environment is discussed in Section 3.9, with particular attention being given to modelling in Section 3.9.1, and evaluation of performability in Sections 3.9.2 to 3.9.4. The performability measure solved for is a distribution of average accumulated reward, which as will be explained, gives an indication of the distribution of average performance. Results are generated for a system of 20 components and presented in Section 3.9.5. The effects on performability of altering reliability and performance data are investigated, and conclusions are presented in Section 3.9.6. Performability can be treated as a measure of the fault-tolerance of a system. In view of these results, in Section 3.10, the concept of a *performability index* is proposed as a means of assessing the relative importance of individual components of a system with respect to aspects of reliability as well as performance. During the course of applying the randomisation technique to a number of test cases, it was found that the distribution of average accumulated reward is not always monotonically increasing. This problem is tackled in Section 3.11 and a suitable check to avoid such discrepancies is also proposed. Another problem was identified as being due to the many-to-one mapping of performance on to rewards; this problem is addressed in Section 3.12. Other common problems affecting evaluation using the *randomisation* technique are described in Section 3.13. One of the major problems consists of the size of the state-space. Techniques of state-space reduction are presented in Section 3.14. In this thesis, the lumping technique is chosen as state-space reduction method, and it is explained in Section 3.15. Evaluation of moments of performability is considered in Section 3.16. Finally, some conclusions are presented in Section 3.17.

3.2 Definition of performability [MEY92] [MEY95]

Performability quantifies, in mathematical terms, a system's *ability to perform* in a designated environment where it is affected by structural changes. The ability to perform is translated into a probability.

The framework in which performability is defined is as follows. Let S denote the *total system* in question, where generally, S is composed of an *object system* C (the computing or communication system being evaluated) and its *environment* E (workload, external faults, etc.). Over a specified *utilisation*, or *observation period*, T, the *performance* of S is represented by a random variable Y that can take values from a set of attainable performance, or accomplishment, levels denoted by A. The period T is an interval of time instants which can be either continuous or discrete and which can be either bounded or unbounded. Given a system, its ability, expressed by probabilities, to achieve different levels of accomplishment within a specified duration of time, is its performability.

For a system S with performance Y taking values in accomplishment set A, the *performability* of S is the probability measure *Perf* induced by Y where, for any measurable set B of accomplishment levels ($B \subseteq A$),

$$\begin{aligned} Perf(B) &= P[Y \in B] \\ &= \text{the probability that } S \text{ performs at a level in } B. \end{aligned}$$

Typically if $A = (-\infty, \infty)$ and $B = [a, \infty)$, then $Perf(B)$ is the probability that S performs at or above level a.

At the lowest level, the dynamics and behaviour of the total system structure are represented by a stochastic process X, referred to as the *base model* of S. The process X is a time-indexed set of random variables given by $X = \{ X_t \mid t \in I \}$, where the *time base* I may be either continuous or discrete, depending on the nature of what is being evaluated. I must include the utilisation period T. For any $t \in I$, the value of the random variable X_t is the state

of the total system S at time t. A base model X coupled with a performance variable Y constitute a *performability model* of S.

3.3 Choice of the performance variable *Y*

Performance here refers to any chosen aspect of total system behaviour relative to which the object system's ability to perform is being measured. The variable Y can be chosen from a wide range of possibilities. These range from a binary-valued variable which is used to indicate whether a delivered service is proper or not, to a multi-valued variable which represents quality of service in terms of a number of accomplishment levels. In the latter case, possible indicators of performance could be measures such as average delay or throughput. When Y is identified with quality of service, the different values of the performance variable are then equivalent to the various degrees of satisfaction experienced by the recipient of the service.

3.4 Performability model construction and solution

An evaluation of performability can be either conducted experimentally via measurements of an actual system, or model-based. The first approach suffers from a number of drawbacks: firstly, taking actual measurements requires the availability of the final system under investigation. Such an approach is not desirable if performability is to be applied at a design stage, where it is more reasonable to have a means of evaluating how the system would perform prior to its implementation. Secondly, measurements need to be gathered over a sufficiently long period of time to be able to capture all possible structural changes which can occur; if a component is associated with a failure rate of once per month, the system needs to be observed over a number of months to allow this event to occur several times. Physical simulation is expensive and time-consuming.

A more sensible approach is to use model-based performability evaluation, which overcomes the limitations of taking actual measurements. The system need not be implemented, and modelling can be done at different levels of detail. It consists of two steps:

1. construction of a performability model for the system and the performance measure in question,
2. evaluation of performability via solution of the model.

Performability model construction involves the specification of the performance variable Y and the determination of the base model X to be solved. Analytical methods, as well as simulation techniques, can be used as modelling tools. Most analytical models that have been used for performability evaluation are based on Markov analysis where a Markov process describes the structural changes in the system resulting from faults and/or repairs. The base model has to be characterised explicitly in some suitable form, which can be used in the solution process. For example, a state-transition matrix can be generated in the case of a continuous-time, time-homogeneous, finite-state Markov process. Less important features of the system can be abstracted away when representing the system in such forms. However, depending on the complexity of the model, it might be mathematically intractable.

If performability is evaluated via simulation techniques, then X refers to the behaviour of some simulation model of S. Computer simulation requires a software modelling of all operations and running the program until a statistical equilibrium is achieved. It is employed especially when the system has to be represented at a fairly high level of detail. The problem that arises here is that in terms of computational and memory requirements the model might be difficult or costly to solve. In both analytical and simulation approaches, approximation techniques are then required to be able to generate an estimate of performability.

Performability model solution consists of the evaluation of values $Perf(B)$ for accomplishment sets B. A knowledge of the probability distribution function (PDF) of Y is generally sufficient to determine such values so that a performability model is regarded as being completely solved once the PDF of Y is determined.

3.5 Markov reward model

A number of performability solution methods are based on *reward models*. A Markov reward model consists essentially of an underlying structure state process and a reward structure. The structure state process, a homogeneous continuous time finite-state Markov chain (CTMC), represents the evolution of the base model X through its different states. This is augmented by a reward structure, with *reward rates* being associated with state occupancies or *reward impulses* attached to state transitions. For example, reward impulses are associated with events such as failures and repairs of individual or groups of components. Such a model is said to be impulse-based. In a rate-based model, reward rates may give an indication of the performance of the system while it is residing in a particular state.

Markov Reward Models lead to a natural combination of the reliability and performance models. The structure state process represents the reliability model whereas the reward structure corresponds to the performance model. This introduces an approximation in that typically the reward rates are constant values [TRI94]. Therefore, it is assumed that for the sojourn time in a state of the reliability model, the system is providing a steady-state performance.

3.6 Review of techniques used to solve for performability

Most of the solution methods use a performability model which is comprised of some form of rate-based Markov reward model along with accumulated reward Y_t as the performance variable in question. Given some value of t, the solution sought is the PDF $F_{Y_t(y)}$, that is, for any accomplishment level y, the probability

$$F_{Y_t(y)} = \mathrm{P}[Y_t \leq y].$$

Gracefully degradable systems can be classified as being either non-repairable or repairable. Generally, applications of non-repairable systems are in critical environments where maintenance is not suitable or uneconomical. Examples of such systems are to be found in spacecraft, satellite, medical, or military environments. Such systems are modelled

using acyclic Markov processes, where they evolve from a particular state of operation to one where performance is worse and so on until a state of complete failure is reached. Cyclic Markov processes are required to model systems which possess some form of repair capability. Once failure of a component takes the system into a state of degraded performance, there exists a reverse transition, indicating some form of repair, which can take it back to its previous state of operation.

In the early 1980's, Meyer [MEY82] obtained a closed form solution of performability for a simple, non-repairable, degradable buffer/ dual-processor system. Later, Goyal and Tantawi [GOY87] developed a closed form expression for the performability in general acyclic structure state processes with monotonic rewards. In particular, they assume that since the system is degradable, $r_i \geq r_j$, if a transition is possible from state i to state j. The variable r_x is the reward associated with state x. This property is referred to as monotonically non-increasing reward rates. Ciciani and Grassi [CIC87] introduced a closed form analytic solution for a class of non-repairable systems, which can be modelled by general acyclic Markov processes, in which the rewards need not be monotonic. The solution, based on use of Laplace transforms, is applied to satellite systems as an example. Donatiello and Iyer [DON87] have also employed Laplace transforms to derive a closed form time-domain solution of performability for such models.

In the case of cyclic models, that are models representing systems possessing some form of repair capability, solutions of the PDF, $F_{Y_t(y)}$, are more difficult to obtain. A number of solutions, applicable to general rate-based Markov reward models, are based on the use of double Laplace transforms which involve transformation of both the time variable t and the accomplishment (accumulated reward) level y. Iyer *et al.* [IYE86] derived a recursive technique for computing the moments of the distribution of average accumulated reward for cyclic Markov reward models. Smith *et al.* [SMI88] and Kulkarni *et al.* [KUL86] have proposed numerical solutions of the double transform systems using inversion. The double transform inversion method works well for small problems. However, it is not suitable for large problems, given that it involves storage, computation and manipulation of the whole generator matrix. The generator matrix characterises the underlying structure-state process. Islam and Ammar [ISL90] derived the PDF of Y_t using Laguerre

transformations and Pattipati *et al.* [PAT89] used linear, hyperbolic partial differential equations to represent the joint distribution between performability and configuration states. They then used finite difference methods to solve for the equations. They also presented a different numerical method, based on the numerical method of lines for solving the partial differential equations [PAT93].

De Souza and Gail [DES89] developed an alternative numerical method based on *randomisation*. The technique is approximate in nature but the accuracy can be specified at the outset. It is applicable to both cyclic and acyclic time-homogeneous Markov processes and can accommodate either impulse-based or rate-based reward models. Although it requires exponential time in the number of distinct rewards, it is still an attractive solution in cases where few reward rates are considered. Donatiello and Grassi [DON91] have obtained an algorithm for evaluating the cumulative performance distribution based on randomisation but with a lower computational complexity. However, the algorithm can be numerically unstable since the coefficients computed in its recursion can have positive and negative signs and are unbounded, which can lead to severe numerical errors and overflow problems.

In this thesis, the randomisation technique, as used by De Souza and Gail, has been adopted to generate the probability distribution function of average accumulated reward and is discussed in greater detail in the following sections.

3.7 The randomisation technique [DES89] [DES92]

The uniformisation, or randomisation, technique is one of the methods used to calculate performability measures. The essence of the technique consists of replacing the continuous time Markov process that represents system behaviour by an equivalent process: a discrete time Markov chain subordinated to a Poisson process. This approach can be used to calculate both transient and steady state solutions.

The *randomisation* technique is reviewed below. The system under investigation is modelled by a homogeneous continuous time Markov process $X = \{X(t) : t \geq 0\}$ governed by a generator matrix **Q** defined on a finite state-space $S = \{a_i : i = 1, \ldots, M\}$.

The generator matrix **Q** is an M x M matrix having the following form

$$\mathbf{Q} = \begin{bmatrix} -q_1 & q_{12} & \cdots & q_{1M} \\ q_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ q_{M1} & \cdots\cdots & & -q_M \end{bmatrix}$$

where q_{ij} = (exponential) rate at which a transition occurs from state a_i to state a_j for $i \neq j$,

and the total rate out of state a_i is $q_i = -\sum q_{ij}$.

Since the state-space is finite, the rates q_i are uniformly bounded by, for example, $\Lambda \geq \max. \{q_i\}$ for $1 \leq i \leq M$. In order to view X as a Markov process for which the rate out of each state is the same parameter Λ, the following construction can be applied. For the i^{th} state, fictitious transitions from a_i back to itself are added so that a transition to state a_j occurs with probability q_{ij}/Λ and a fictitious self-transition occurs with probability $1-(q_i/\Lambda)$. The basic structure of X is preserved but the transition rates for all states are now identical.

Thus, the Markov process X can be thought of as a discrete time Markov chain subordinated to a Poisson process in the following sense. Consider the discrete time Markov chain $Z = \{Z_n : n = 0, 1, ...\}$ with state-space S and with transition matrix $\mathbf{P} = \mathbf{Q}/\Lambda + I$ and let $N = \{N(t) : t \geq 0\}$ be a Poisson process with rate Λ that is independent of Z. From the above discussion, $X(t)$ can be interpreted as $Z_{N(t)}$ for $t \geq 0$. In other words, the transition times of X occur according to the Poisson process N, while the transitions themselves are governed by the discrete time chain Z according to the transition matrix **P**.

Having carried out the transformation, numerically stable algorithms for the calculation of transient and steady state distributions of various measures can be derived [GRA77]. For example, if a measure $M(t)$ is to be evaluated, when conditioning on the number n of (Poisson) transitions in a period of length t,

$$M(t) = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} M(t|n) . \qquad (3.1)$$

More specifically, if $M(t)$ represents the transient state probability distribution $P(t)$ $= <P_1(t), \ldots, P_M(t)>$, where $P_i(t)$ represents the probability that X is in state a_i at time t under the initial distribution $P(0)$, that is $P(t) = \mathrm{P}[X(t) = a_i]$,

$$\text{then, } P(t) = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \pi(n) \qquad (3.2)$$

where $\pi(n) = <\pi_1(n), \ldots, \pi_M(n)>$, and $\pi_i(n)$ is the probability that the chain Z is in the i^{th} state at time n, that is $\pi_i(n) = \mathrm{P}[Z_n = a_i]$. $\pi(n)$ satisfies the recursion $\pi(n+1) = \pi(n)\mathbf{P} = \pi(0)\mathbf{P}^n$ with initial distribution $\pi(0) = P(0)$.

Randomisation can also be used to calculate performability measures of Markov reward models.

3.8 Use of randomisation to calculate transient performability measures [DES92]

Transient performability measures can be calculated using uniformisation over a finite observation period $(0, t)$. Transitions of the uniformised chain split the time period of interest into intervals during which the process X remains in a particular state. The various performability measures depend on certain states and/or transitions. The methodology described below is based on "colouring" intervals corresponding to these states or transitions. Each interval is associated with colours, or rewards, which are usually based on

the state of the system during that interval, or on the transition at the start of the interval. Intervals with identical rewards are assigned the same colour, and randomisation is used to calculate the resulting measures which are defined in terms of the colours.

There are K+1 different rewards with $1 = \rho_1 > \ldots > \rho_{K+1} = 0$ associated with states or transitions (pairs of states) of the Markov process X. Let $M(t)$ be the measure of interest which has to be calculated. If n transitions of X occur during $(0, t)$, the observation period is divided into $n+1$ intervals. Assume that there are k_l intervals of colour l (with reward ρ_l) for $l = 1, \ldots, K+1$. The vector $\boldsymbol{k}$ is defined, where $\boldsymbol{k} = < k_1, \ldots, k_{K+1} >$ which indicates the number of intervals corresponding to each reward. This is often referred to as a colouring $\boldsymbol{k}$. $\| \boldsymbol{k} \| = k_1 + \ldots + k_{K+1}$. Conditioning on n and $\boldsymbol{k}$ gives

$$M(t) = \sum_{n=0}^{\infty} e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \sum_{\|k\|=n+1} \Omega[n,k] M(t|n,k) \qquad (3.3)$$

where $\Omega[n, k]$ = P[colouring $\boldsymbol{k}$| n transitions]

and $M(t, n, \boldsymbol{k}) = M(t) \mid n$ transitions, colouring $\boldsymbol{k}$.

This is the main equation which is used to calculate the various performability measures of interest.

3.9 Application of the randomisation technique

3.9.1 Modelling of failure/repair process in communication networks using the Markov reward model

Modelling of a communication network can be done at any level of detail, from an overall view of the network down to an intricate description of all possible events which can affect the system. Similarly, all components, or just a selection, can be included in the model depending on their relevance to the particular measure that needs to be evaluated.

A Markov reward model is used to model the failure/repair process in a communication network. In general, each state of the Markov chain represents a particular configuration of the network which corresponds to a set of operational components. Transitions between the states occur on the failure, or repair, of particular components. Every state is further augmented by a reward which gives an indication of the performance of the network when it is operating in that particular configuration. Performance could refer to average network delay, average throughput of packets, or any other measure of assessing the network; with it being understood that the better the performance, the higher the reward. If performance is identified with a measure such as average delay across the system, then performance and reward are, in a sense, inversely proportional to each other: a low delay is an indication of good performance and is associated with a high reward. When failure occurs, the delay increases, which represents a deterioration of performance, and the corresponding reward decreases. If performance is expressed in terms of throughput, performance and reward are directly proportional to each other, and reward increases with an increase in throughput.

The mapping from performance to reward is illustrated in Figure 3.1 below. Performance is assumed to be identified with a measure such as average delay in a manner that the lower the delay, the better the performance, and hence the higher the reward. Assignment of rewards to individual states is done based on the steady-state performance of the network in that configuration. It does not take into account the transient period when there is a switch of states caused by a failure or repair.

State a_0 corresponds to the fully operational configuration, and any state a_i, where $i > 0$, represents a state of failure of one or more components. An up transition from state a_i to state a_j, where $j > i$, represents failure of a component, whereas a down transition from state a_j to state a_i represents a repair action. While operating in state a_0, the system is assumed to be delivering a level of performance denoted by *Ref.* Rewards vary between 0 and 1 so that when the system is fully functional, it is associated with a reward equal to 1. Upon the occurrence of an element failure, the level of performance, that is the delay, increases with a corresponding decrease in reward.

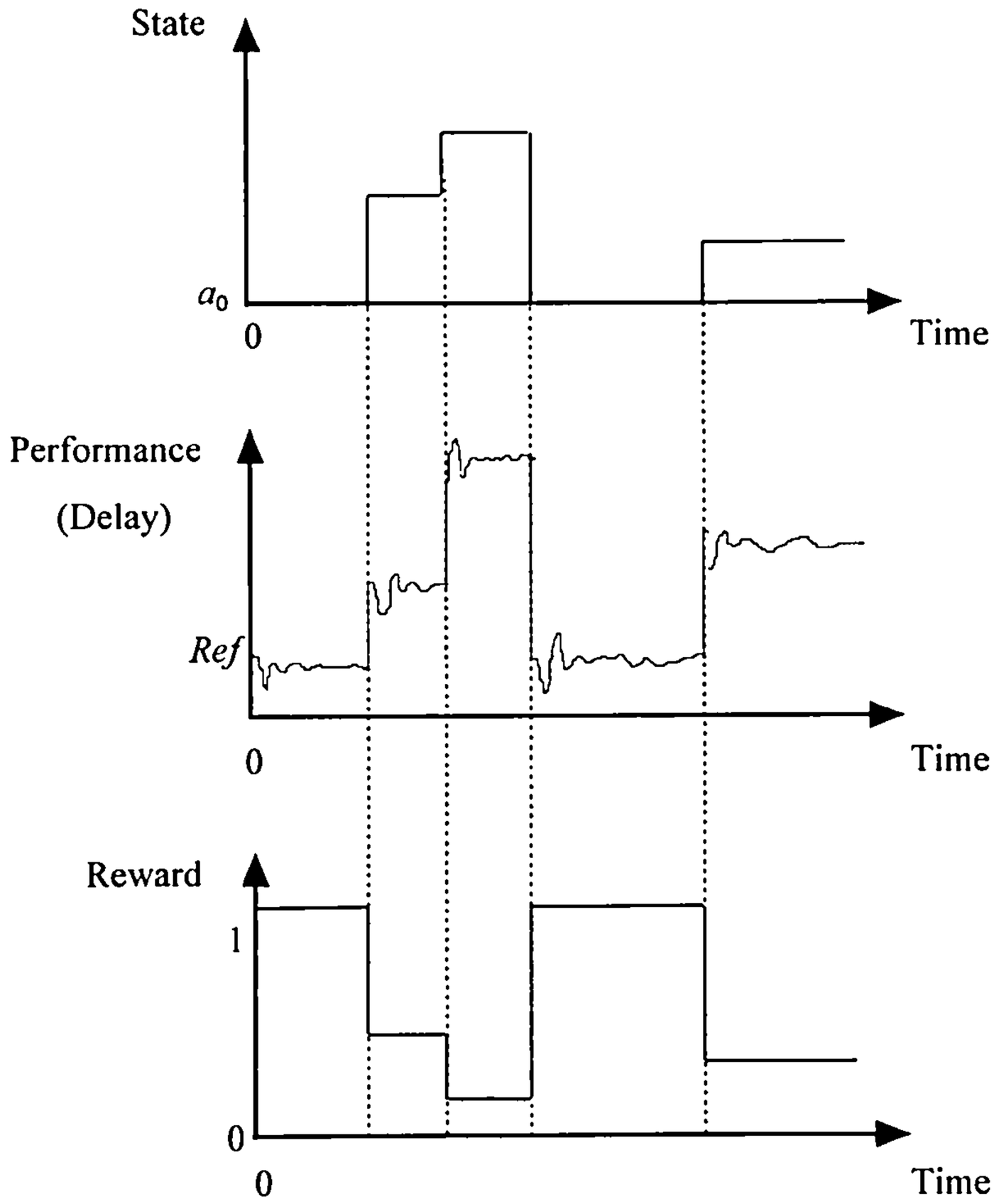


Figure 3.1 - Mapping from performance on to reward

The 2-node network shown in Figure 3.2 is used as an example to show modelling at different levels of detail.

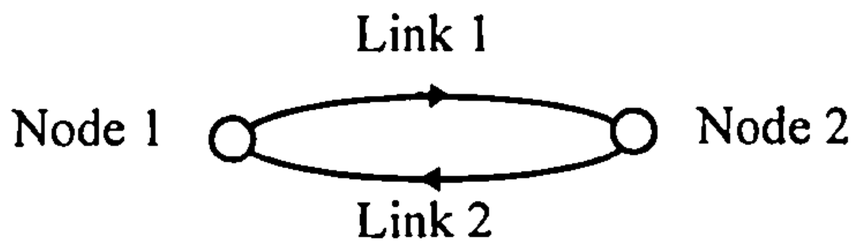


Figure 3.2 - 2-node network

The Markov reward model in Figure 3.3 represents an overall view of the network with respect to link failures, where a link is considered to be either operational or not.

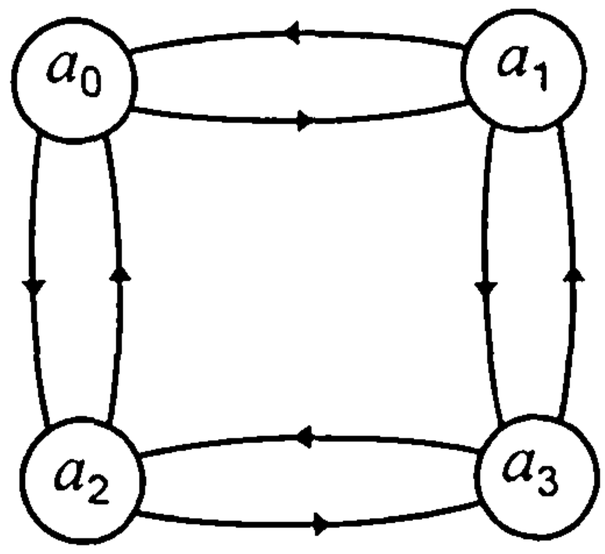


Figure 3.3 - Model for 2-node network

State a_0 corresponds to the network when all links are operational; and state a_3 represents the system when both links have failed. States a_1 and a_2 then correspond to configurations where links 1 or 2 respectively have failed. A transition from the fully operational state to one of failure indicates failure of the particular component, whereas a reverse transition represents repair of the component. A transition from an intermediate state of failure, that is state a_1 or a_2, to state a_3 occurs upon failure of the link which was operational. For example, a transition from state a_1 to state a_3 indicates that link 2 has failed; a reverse transition from state a_3 to state a_1 would correspond to repair of that link.

Figure 3.4 represents a possible model of a particular link if modelling is carried out at a higher level of detail. State a_0 corresponds to the configuration where the link is up, operating at full capacity and performance is acceptable, and state a_3 is the one where the link has completely failed. States a_1 and a_2 are states of degraded performance where the link is physically working but is operating at a fraction of its capacity so that the delivered performance is not within acceptable limits. State a_2 represents more degradation than state a_1. Here again, transitions from a given state to one where there is more degradation, that is from state a_i to a_j where $j > i$, represent a form of failure, whereas transitions in the opposite direction indicate some form of repair or reconfiguration.

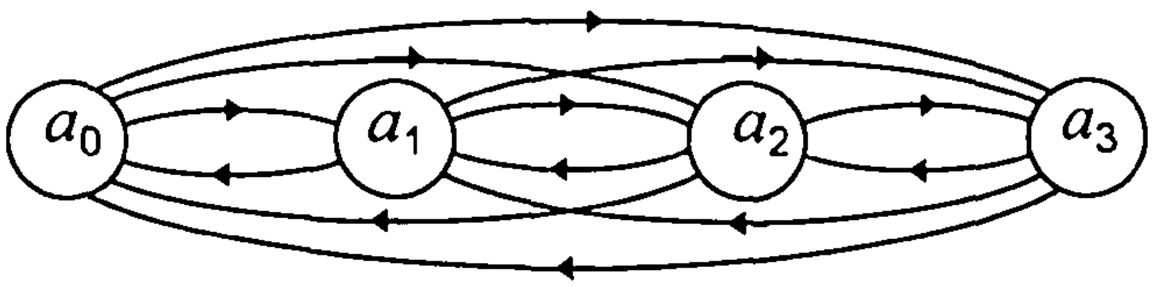


Figure 3.4 - Model for a single link

In a general model, every possible state as dictated by failure and repair of components is included. However, it is fairly obvious that by allowing multiple failures, and different levels of failure and/or partial repair, a huge state-space will result, even for a relatively small network. This is demonstrated from the given example where if every link was modelled at the level of detail shown in Figure 3.4, the overall network model would need to incorporate all these states and become a lot more complex. The size of the state-space is a matter of concern in evaluation of performability. In an attempt to minimise the size of the state-space, modelling of the network can be based on the following assumptions.

1. Only one component is allowed to fail at any one time.
2. Failures are judged to be sufficiently rare so that once a component has failed, no other component fails prior to repair of that component. Repair rates are much higher than failure rates.
3. Node failure causes all links connected to it to go down, but these are not considered as link failures. Repair of the node brings them back to operation.

Based on these assumptions, a network of M components is modelled by a Markov chain of M+1 states, states a_0 to a_M, as shown in Figure 3.5. State a_0 represents the fully operational state where all components of the network are functioning. State a_i where $1 \leq i \leq M$ represents the configuration of the network when component i fails. Given that only single failures are allowed, transitions are allowed only from state a_0 to other states and vice versa. A transition from state a_0 to state a_i indicates failure of component i, whilst a transition from state a_i to state a_0 shows repair of component i. A repair always takes the network back to its fully operational state.

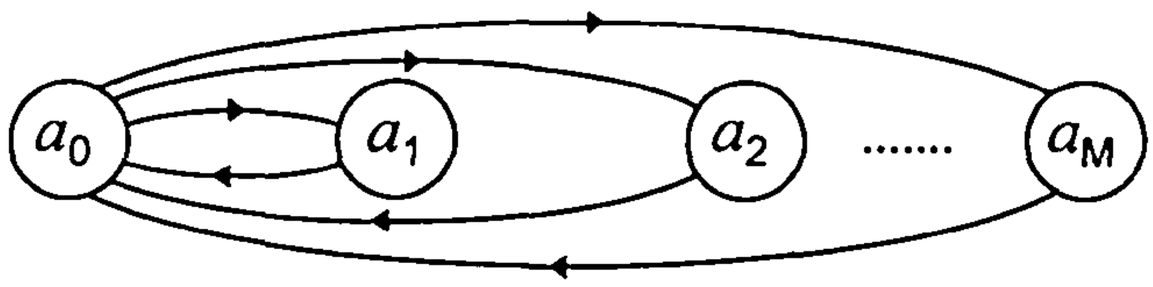


Figure 3.5 - General network model

Hence, with the structure state process representing the failure/repair process in the network and the rewards representing performance measures, the Markov reward model becomes the base model X which can be used for evaluation of performability measures.

Using the randomisation technique, the cyclic homogeneous continuous time Markov chain can be transformed into a discrete time Markov chain subordinated to a Poisson process. The corresponding transition matrix can then be derived, and performability can be evaluated.

3.9.2 Application of the randomisation technique to calculate the PDF of the average accumulated reward

The performability measure sought is the probability that the average performance of the network is less than or equal to a reference value, thereby leading to the evaluation of the probability distribution function of the average performance. Rewards which represent performance levels are associated with the states instead of transitions so that for each state a_j, there is a corresponding reward r_j. Given the mapping from performance on to reward, a distribution of average accumulated reward gives an indication of the distribution of average performance. If the mapping from performance to reward is one-to-one, then an average performance maps exactly on to an average accumulated reward so that their respective distributions map on to each other. In general, the computational requirements of the randomisation technique increase exponentially with the number of rewards. Therefore in general, a range of performance levels is mapped on to one reward. The distribution of average accumulated reward then gives an indication of the distribution of average performance but they do not map exactly onto each other. The higher the number of rewards, the closer the relation between the two distributions.

It should be noted that the relation between a distribution of average performance and a distribution of average accumulated reward depends on the choice of performance measure. If performance is taken to be a measure such as network delay, an inverse mapping exists between performance and reward, and hence the corresponding distributions. If performance is identified with a measure such as throughput, then a direct relation holds.

The measure, $M(t)$, which will be evaluated is a distribution of average accumulated reward $ACR(t)$.

Given that the operation of the network has been observed over a period of t units, for example 30 days, a sample path of the Markov chain can be obtained. The sample path depicts the network changing states as its components fail and are repaired. Assuming that n transitions have occurred with a colouring $\boldsymbol{k}$, total accumulated reward averaged over t is given by:

$$\textit{average accumulated reward, } ACR(t) = \frac{1}{t}\sum_{l=1}^{K+1}\rho_l\zeta_l \qquad (3.4)$$

where t is the observation period,

ρ_l is reward l,

and ζ_l is total duration of time for which reward is equal to ρ_l.

For this sample path, a colouring vector $\boldsymbol{k}$ can be derived. Recall that a colouring $\boldsymbol{k}$ means that there are k_1 intervals associated with reward ρ_1, k_2 intervals with reward ρ_2 and so on. There are K+1 different rewards with $1 = \rho_1 > \ldots > \rho_K > \rho_{K+1} = 0$.

$M(t)$ is given by $P[ACR(t) \leq r]$ and according to equation (3.3), is calculated in the following way:

$$P[ACR(t)\leq r] = \sum_{n=0}^{\infty} e^{-\Lambda t}\frac{(\Lambda t)^n}{n!}\sum_{\|k\|=n+1}\Omega[n,k]\,P[\mathrm{ACR}(t)\leq r|n,\ k] \qquad (3.5)$$

where $\Omega[n, k]$ is the probability that a colouring $\boldsymbol{k}$ is obtained given n transitions.

3.9.3 Evaluation of the conditional distribution P[$ACR(t) \leq r \mid n, k$] [DES92]

For notational convenience, it is assumed that every colour, or reward, appears at least once in the vector $\boldsymbol{k}$. If this is not the case, certain terms in the sum are not present and re-labelling has to be done.

Suppose that n transitions of the randomised chain occur at times $0 < \tau_1 < ... < \tau_n < t$ during the observation period $(0, t)$. These transitions split the period $(0, t)$ into $n+1$ intervals with lengths $Y_1, \ldots, Y_{n+1}$. During the i^{th} interval, the process X remains in a particular state Z_{i-1}, which depends only on the underlying discrete time Markov chain Z. Furthermore, the transitions are governed by a Poisson process of rate Λ, so that the transition times τ_i are distributed as the order statistics of n independent and identically distributed random variables uniform on $(0, t)$. Let $U_1, \ldots, U_n$ be iid variables on $(0,1)$ and let $U_{(1)}, \ldots, U_{(n)}$ be their order statistics. Then, τ_i can be identified with $tU_{(i)}$, so that $Y_1 = tU_{(1)}$, $Y_i = t(U_{(i)} - U_{(i-1)})$ for $i = 1, \ldots, n$ and $Y_{n+1} = t(1 - U_{(n)})$. Although the Y_i are dependent random variables, given that their sum is the fixed number t, it is well-known that they are exchangeable [ROS83]. The joint distribution of the Y_i is invariant under any permutation of $1, \ldots, n+1$.

Owing to the exchangeability of the Y_i and the independence of N and X, it can be assumed that the first k_1 intervals are associated with reward ρ_1, the next k_2 intervals with reward ρ_2, and so on. Defining

$$n_j = \sum_{l=1}^{j} k_l \quad \text{for } j = 1, ..., K, \text{ the following can be derived.}$$

$$\zeta_1 = tU_{(n_1)}$$

$$\zeta_j = t\left(U_{(n_j)} - U_{(n_{j-1})}\right) \text{ for } j = 2,...,K$$

$$\zeta_{K+1} = t\left(1 - U_{(n_K)}\right)$$

$$\textit{Average accumulated reward } ACR(t) \mid \boldsymbol{n}, \boldsymbol{k} = \sum_{j=1}^{K}\left(\rho_j - \rho_{j+1}\right)U_{(n_j)}, \qquad (3.6)$$

and hence the conditional distribution is

$$P[ACR(t) \le r \mid \boldsymbol{n}, \boldsymbol{k}] = \mathrm{P}\left[\sum_{i=1}^{K}\left(\left(\rho_j - \rho_{j+1}\right)U_{(n_j)}\right) \le r\right] \qquad (3.7)$$

The conditional distribution is evaluated using Weisberg's result [WEI71] as shown below in which m is defined to be the largest index i such that $r \leq \rho_i$. Weisberg's result is given by

$$P[ACR(t) \leq r \mid n, k] = 1 - \sum_{i=1}^{m} \frac{g_i^{(k_i-1)}(\rho_i)}{(k_i - 1)!} \tag{3.8}$$

where $g_i^{(k)}$ is the k^{th} derivative of the function

$$g_i(x) = \frac{(x-r)^n}{\prod_{l=1, l\neq i}^{K+1} (x-\rho_l)^{k_l}} \tag{3.9}$$

The derivatives of the functions $g_i(x)$ may be evaluated recursively in a straightforward manner [WEI71] [DES89].

3.9.4 Evaluation of the conditional probability Ω(*n*, *k*) [DES92]

Let $\Omega(n, k)$ be the probability that the colouring of intervals is equal to vector k, given that n transitions have occurred. The colouring of intervals is based on the state of the process X (equivalently the state of the chain Z). For each state a_j, $c(j)$ is defined to be the colour associated with it, that is $r_j = \rho_{c(j)}$ where ρ is the reward structure.

Let $w_i[n,k]$ be the probability that there are k_l intervals with reward ρ_l ($l = 1, \ldots,$ K+1), where there are n transitions of Z, and $Z_n = a_i$. Also let the vector $w[n, k] = \langle w_0[n, k], w_1[n, k], \ldots, w_M[n, k] \rangle$ for a system of M components. Each element of the vector is evaluated recursively using equation (3.10).

$$w_j[n,k] = \sum_{i=0}^{M} w_i\left[n-1, \underline{k} - 1_{c(j)}\right] \mathbf{P}_{ij} \tag{3.10}$$

where $1_{c(j)}$ is a vector of length K+1 with 0 in all entries except in position $c(j)$ where the entry is 1 and $\mathbf{P}_{ij}$ is the $i-j^{th}$ entry of the transition matrix $\mathbf{P}$ for the discrete time Markov chain Z.

The base conditions of the recursion are, when n is equal to 0,

$$w_j[0,\boldsymbol{k}] = \pi_j(0) \text{ if } \boldsymbol{k}_{c(j)} = 1 \text{ and all other entries of the } \boldsymbol{k} \text{ vector are equal to 0}$$
$$= 0 \quad \text{otherwise.} \tag{3.11}$$

$\pi(0)$ is the initial probability distribution of the states at time 0, so that $\pi_j(0)$ is the probability that the system starts in state j at time 0.

$$\text{Then } \Omega[n, \boldsymbol{k}] = \| w[n, \boldsymbol{k}] \|. \tag{3.12}$$

The number of vectors $w[n, \boldsymbol{k}]$ grows combinatorially with the number of performance levels but the recursion can be organised in such a way as to minimise the computation involved. An increase in the speed of evaluating $\Omega[n, \boldsymbol{k}]$ has been obtained at the expense of an increase in memory required to store intermediate results.

3.9.5 Results

The methodology described to calculate the transient performability measures using uniformisation, or randomisation, can be applied to various systems with different numbers of components in order to determine the probability distribution function of the average accumulated reward; thereby giving an indication of the probability distribution function of the average performance. In this section, the performability of a system of 20 components is obtained. The effects of varying aspects of the reliability and performance data are also investigated.

The reward structure will be considered to consist of three reward rates, 1, 0.5 and 0. Depending on the value of the performance level in each state, a corresponding reward is associated with it. The better the performance level, for example, low delay or high throughput, the higher the reward. Note that the mapping performance on to reward is not a one-to-one mapping. Given the large span of performance levels and the few rewards, a single reward represents a range of performance levels rather than just one. Furthermore, it is assumed that the fully operational state is associated with the highest reward, and all states of failure are associated with lower rewards.

Ideally, the probability distribution function of the average accumulated reward of a system is expected to be a step function $\mathbf{u}(r - 1.0)$ which basically means that the probability that the average accumulated reward is less than 1.0 is 0 and the probability that it is equal to 1.0 is 1. The system is analysed under two different sets of conditions. In results A, the effects of varying the failure and repair rates with respect to the original distribution function and the ideal one are investigated. In results B, the performance levels attached to each state are changed, and resulting effects on the performability distribution are analysed.

3.9.5.1 Results A

The system is investigated with respect to changes in the failure and repair rates. This leads to a change in the transition matrix but the reward structure associated with the Markov chain remains unmodified. Figure 3.6 shows the distribution curves obtained for the cases where the failure rates, or the repair rates, are doubled with all the performance levels being constant.

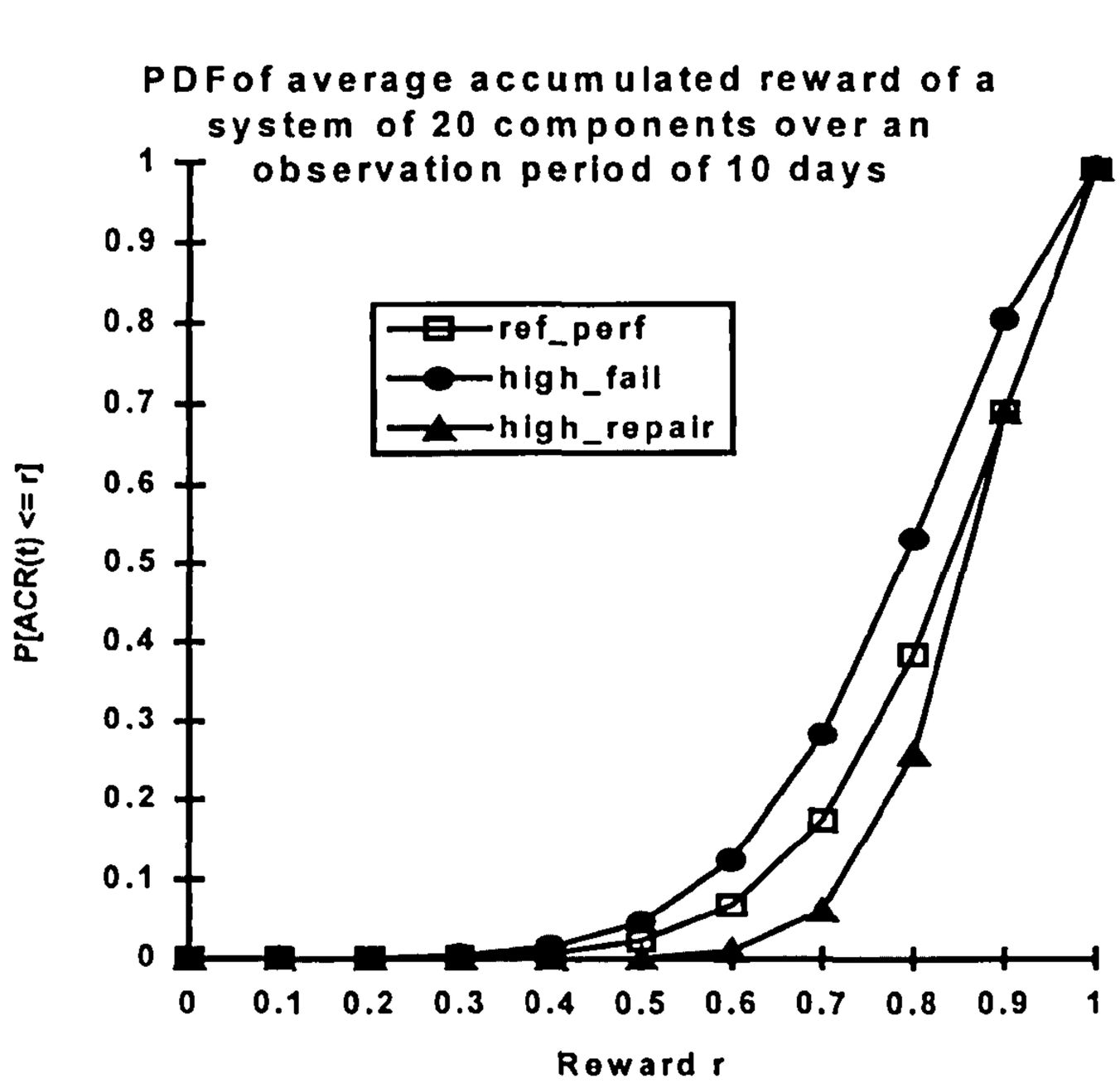


Figure 3.6 - Results when rates of structural changes are varied

ref_perf : *PDF of average accumulated reward of reference system of 20 components.*

high_fail : *PDF of average accumulated reward of similar system except that failure rates of the components are doubled.*

high_repair : *PDF of average accumulated reward of similar system except that repair rates of the components are doubled.*

The direct implication of more frequent failures is that the system becomes less reliable so that the average performance deteriorates when taking into consideration the behaviour of the system over the observation period. This is indicated by a shift of the probability distribution profile *high_fail* away from an ideal distribution. From a mathematical point of view, on increasing the failure rates λ_i, the probability of the system remaining in the fully operational state, which is given by $1-\sum_{all\ i}\lambda_i / \Lambda$, is reduced. The reward accumulated in that state, which is associated with the highest reward, is reduced. Higher repair rates tend to make the system perform better, which is shown by a shift of the distribution *high_repair* towards the ideal step function. Once the system is in a failure state a_i, the probability that it remains in that state, given by 1 - μ_i/Λ, is reduced so that it has a higher probability of going back to the fully operational state where reward can be accumulated at a higher rate.

3.9.5.2 Results B

The system is analysed with respect to changes in the performance levels attached to the states of the Markov chain which cause the reward structure to be modified. Failure and repair rates remain constant so that the transition matrix does not change. Figure 3.7 shows the results obtained.

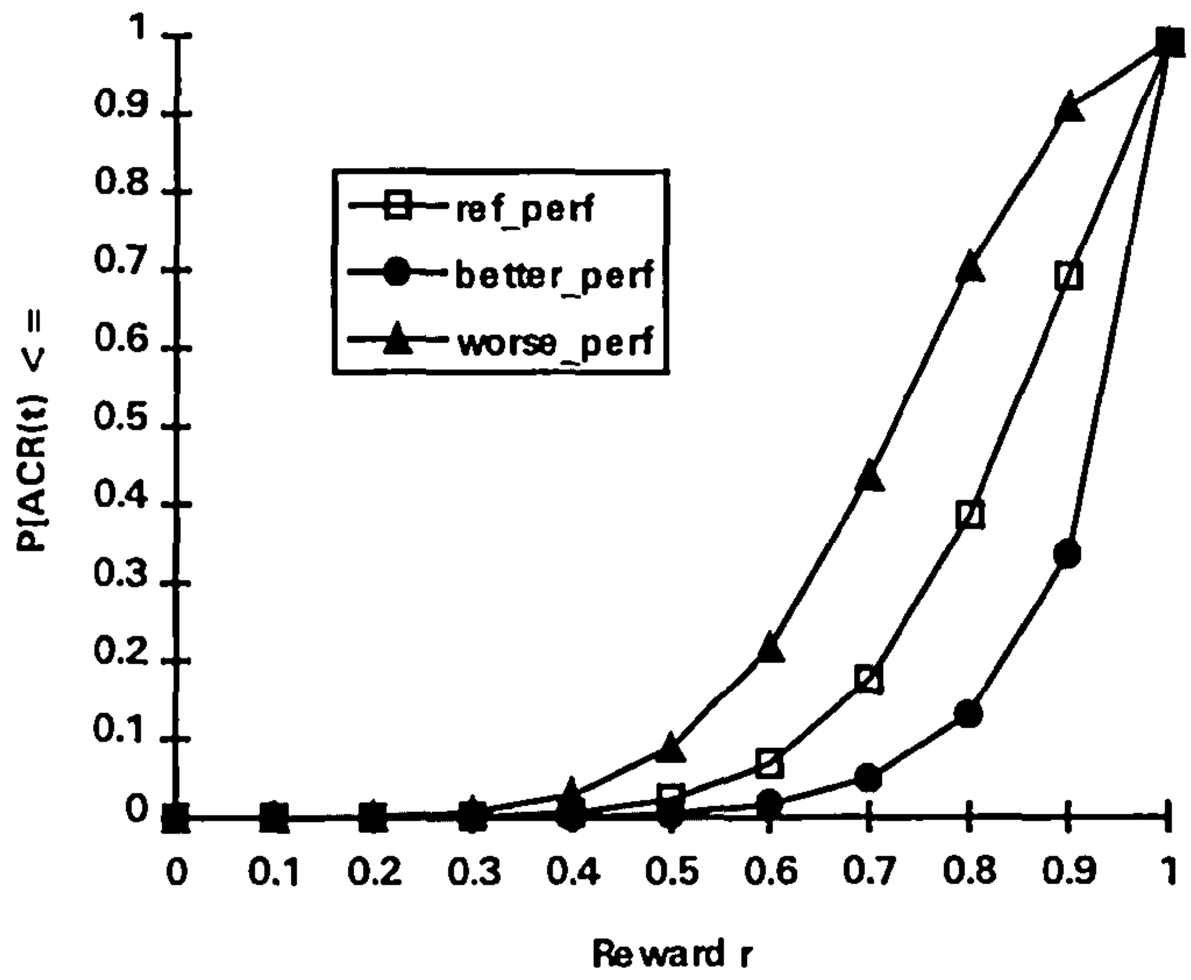


Figure 3.7 - Results when performance data is varied

ref_perf : *PDF of average accumulated reward for reference system of 20 components.*

better_perf : *PDF of average accumulated reward for same system except for better performance levels in some states.*

worse_perf : *PDF of average accumulated reward for same system except for worse performance levels in some states.*

In *better_perf*, all performance levels are decreased and assuming the performance variable is identified with a measure such as the average delay across the system, the resulting system is expected to show an improved performance. Better performance levels in failure states mean less degradation under conditions of failure and, as expected, the performability distribution shows a clear shift towards the ideal one. Conversely, the distribution moves further away when the performance levels associated with the states are increased, as shown by *worse_perf*.

3.9.6 Means of assessing the fault-tolerance of a system

A fault-tolerant system is one where failures lead to a degradation of performance without always causing complete shutdown or unavailability. In a network context, the fault-tolerance of a network lies in its ability to withstand failures, in other words, its ability to perform in spite of failure of its components, that is, its nodes and links. Fault-tolerance is therefore directly dependent on reliability and performance.

Results A have shown the effect of varying the rate of structural changes on average performance. The system appears to be performing better when repair rates are increased. Note that *perform better* in this context does not mean that the performance level associated with each state changes but that the system's ability to perform, taking into account the structural changes, has improved. In that sense, this can be taken as an indication of an increase of the fault-tolerance of the system. In the same way, higher failure rates lead to a deterioration of the fault-tolerance.

Results B show the effect of varying the reward structure. On the one hand, performance levels are increased and, as expected, the system shows worse performance, given that performance is identified with a measure such as delay. On the other hand, a decrease in the performance levels leads to a better performing system. Here, under same conditions of failures and repairs, a system with better performance levels in its degraded states obviously shows a higher degree of fault-tolerance via less degradation caused by failures. Similarly, if performance levels are made worse, it means that failures have a more drastic effect on the performance of the system, hence a deterioration of its ability to withstand failures.

On the whole, in a communication network context, using a Markov reward model to represent a network enables modelling of the failure/repair process of its components without forgetting its effects on the performance of the network. Further analysis using the randomisation technique generates the probability distribution of the average performance which effectively is a measure of the fault-tolerance of the system being analysed, that is the network.

3.10 Concept of performability index

For a given system, every constituent component has its relative importance in terms of how much it contributes to the overall operation. The system could be a network of computers connected by communication links or a single computer comprising processors, memory modules and so on.

One way of determining this quantity for each component is to assess the performance of the whole system when that particular element is not working. The component whose failure causes most degradation in performance is considered to be the most important. Such a measure excludes any aspect of reliability of the components, that is, how often they fail, or how quickly they can be repaired. A more reasonable assessment of the importance of individual components would take into consideration their failure rates, mean repair times as well as the degree of performance degradation resulting from their failure.

Such an approach gives rise to a number of questions as to how to decide which component is more important. Is it one which fails frequently without causing much degradation in performance, or is it one which fails infrequently, but which results in a drastic deterioration in performance? Is a component which is associated with a drastic degradation in performance on failure, but which can be repaired quickly, more significant than one which cannot be brought back to operation as fast, but does not cause so much degradation on failing? These questions are of relevance particularly at a design stage, for instance when decisions regarding where to incorporate redundancies need to be taken. Suppose that the various components of system can be coupled with a redundant component which takes over if the component fails. The questions which need to be addressed are which component is more important, and where would it be more beneficial to add a redundant component. Such questions should be answered with a view of the system over a period of time, as it evolves through various states due to failures and repairs of its components. By assessing the various parts of a system with respect to their failure rates, mean repair times, and degree of performance degradation caused on their failure, they are being appraised with respect to their contribution to the fault-tolerance of the overall system.

Based on the premise that a distribution of performability gives a measure of the fault-tolerance of a system, in this chapter, the *performability index* is proposed as a concept which includes aspects of both performance and reliability in determining the significance of a component in a system. It is a measure associated with each component, which is expected to provide an indication of its relative importance in respect of its failure rate, mean repair time, and the degree of performance degradation resulting from its failure. There are two ways of assessing how important each component is in maintaining the average performance of the whole system over a period of time. This leads to two proposed definitions of the performability index.

3.10.1 Performability index : Definition 1

In the first definition, each component is considered on its own; and the remaining components are ignored. The index is evaluated as $P[ACR(t) \leq r_p]$ for each component, where r_p is a constant value for an observation period t. It should be noted that $ACR(t)$ corresponds to the average accumulated reward, and with a mapping of performance on to reward, such a measure is equivalent to average performance. The model consists of two states, one corresponding to the whole system with the element being operational and one where it has failed. The performance measure associated with each state is equivalent to the performance of the whole system. Recalling that the ideal distribution of $ACR(t)$ is a step function $\mathbf{u}(r - 1.0)$, at reward equal to r_p, the component with the highest index is judged to be the most important.

Given the current definition of the performability index, it is applicable only to the single failure/ total repair scenario. In essence, the system is being decomposed and the contribution of each component evaluated on its own. The performability index thus defined gives an accurate indication of significance in the case where multiple failures, or partial repair are not allowed.

In the case of single failure/ total repair scenario, a typical sample path would be similar to the one shown in Figure 3.8. Here, performance is identified with a measure, such as delay, so that upon failure, the value of performance increases. In the sample path shown,

it is assumed that there are three components which can fail one at a time, and performance degradations are indicated by *Perf1*, *Perf2* and *Perf3* respectively. In the figure, *Ref* denotes the performance of the system when it is fully operational.

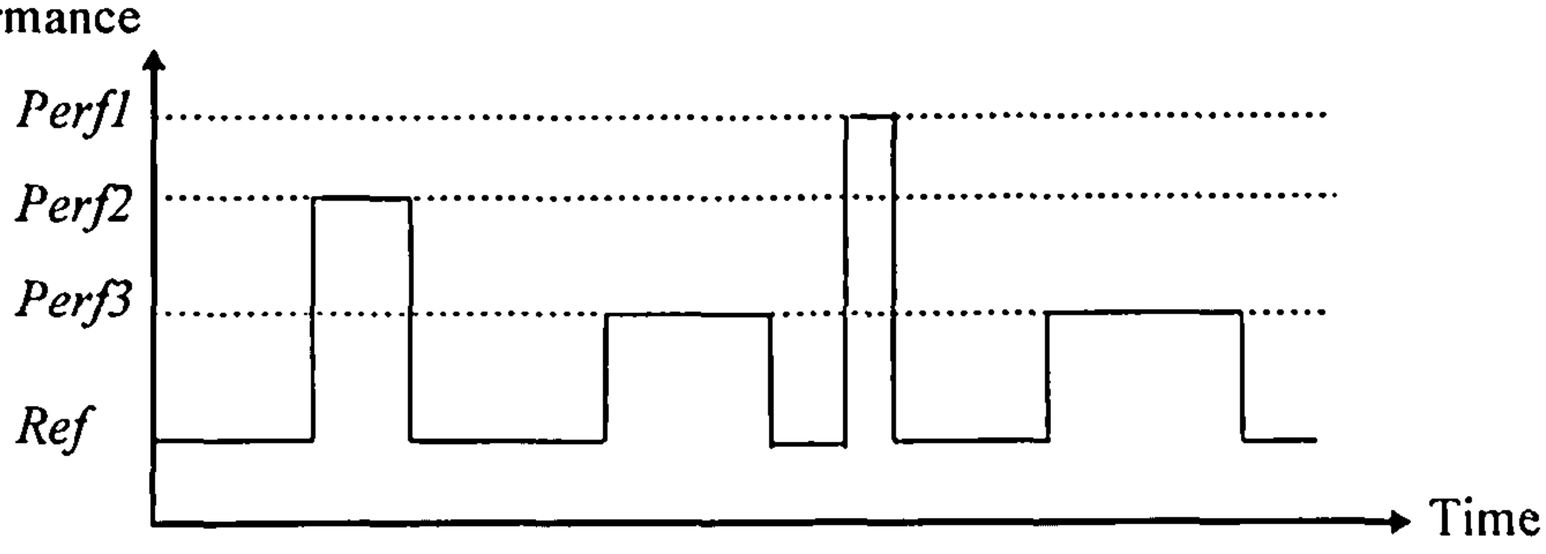


Figure 3.8 - Typical sample path for a system of three components

Assuming that failures are independent of each other, the contribution of the various components to the above sample path can be decomposed into three sample paths, as shown below in Figures 3.9(a), 3.9(b) and 3.9(c).

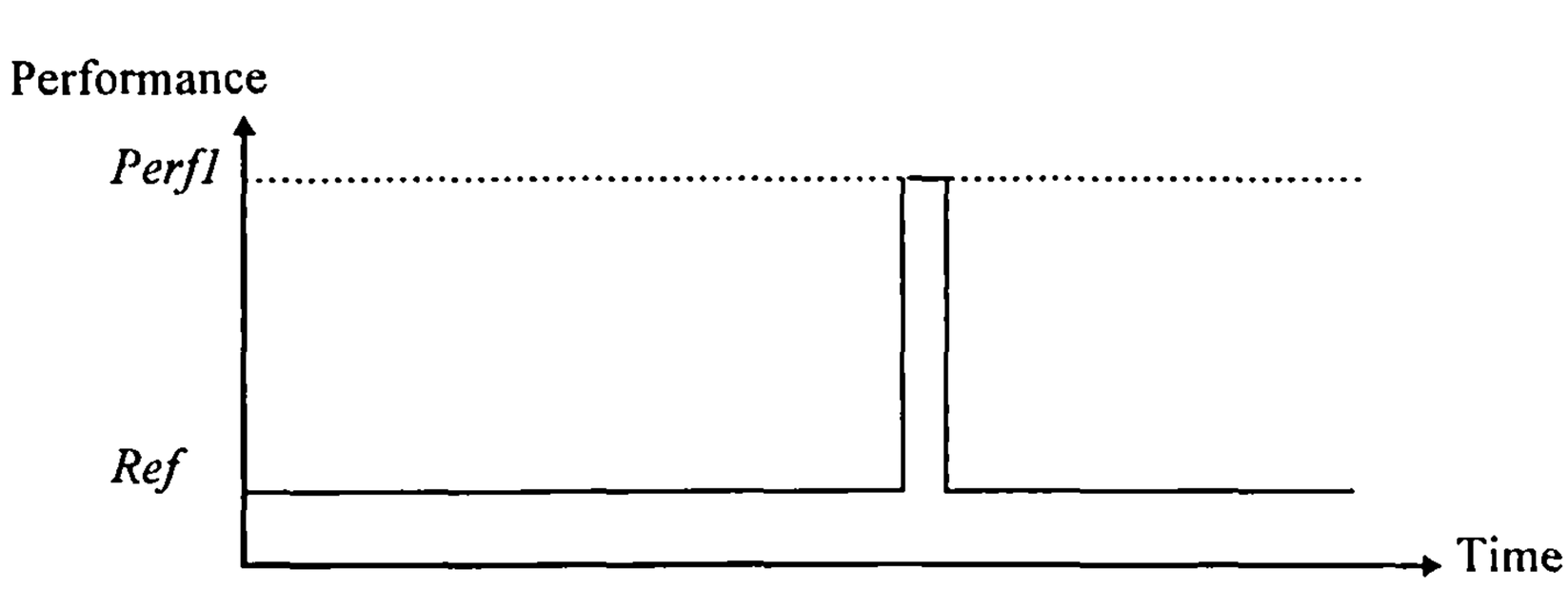


Figure 3.9(a) - Sample path for component 1 for Definition 1

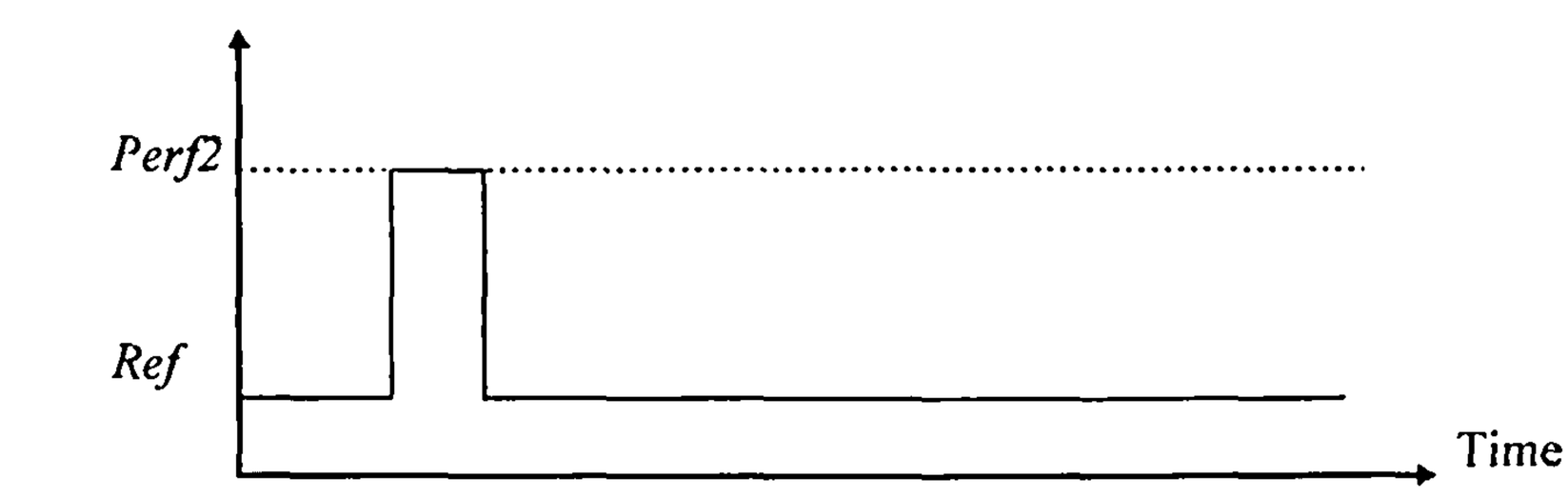


Figure 3.9(b) - Sample path for component 2 for Definition 1

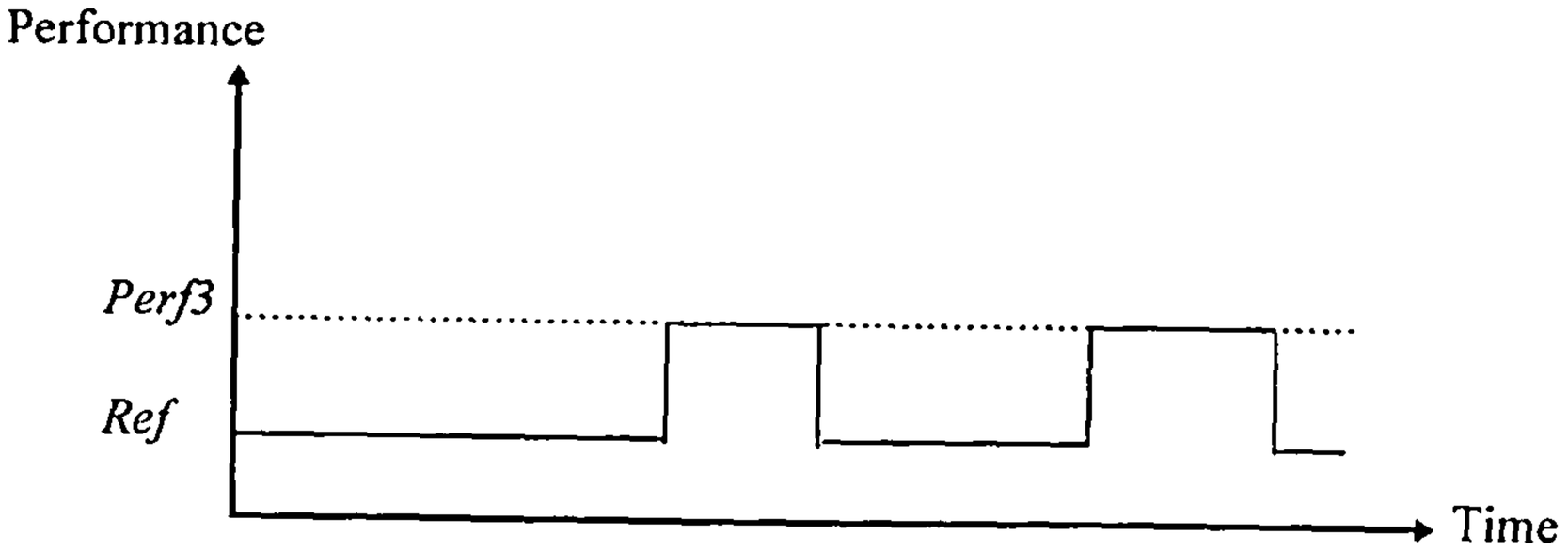


Figure 3.9(c) - Sample path for component 3 for Definition 1

Determining the value of the performability index for each element consists of evaluating the performability at a specific point r_p for each one; in other words, determination of the index is based on the decomposed sample paths.

3.10.2 Performability index : Definition 2

In the second definition, the importance of a component is assessed with respect to the degree of improvement that would be achieved in performability if it was that the component is totally reliable. The greater the improvement in performability as compared with that obtained when all components are allowed to fail in turn, the more important the component is. With this definition, the lower the index, the greater the improvement in performability. Note that definition 1 only includes data relating to the component whose significance is being evaluated whereas definition 2 takes into consideration the operation of the other components in the system. The index is still evaluated as $P[ACR(t) \leq r_p]$, but uses the whole state-space.

Consider again the sample path of the system of three components, shown in Figure 3.8. In evaluation of the performability index, given the above assumption, the corresponding sample paths for each component are as shown in Figures 3.10(a), 3.10(b) and 3.10(c).

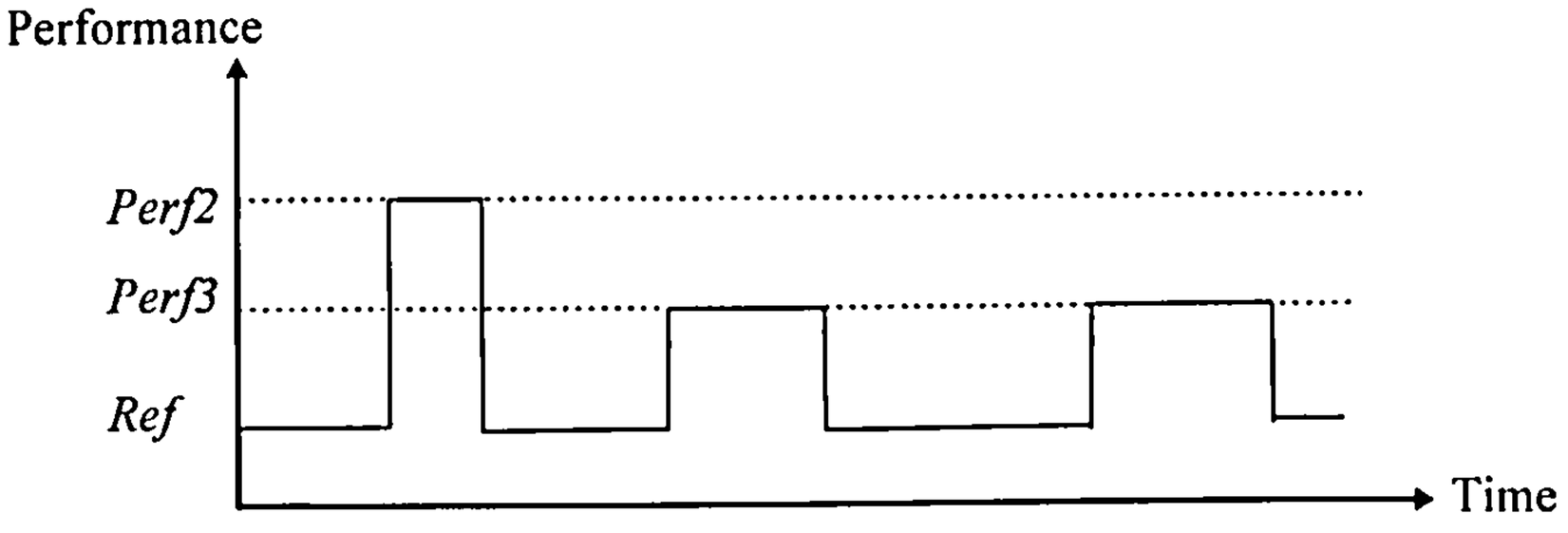


Figure 3.10(a) - Sample path for component 1 for Definition 2

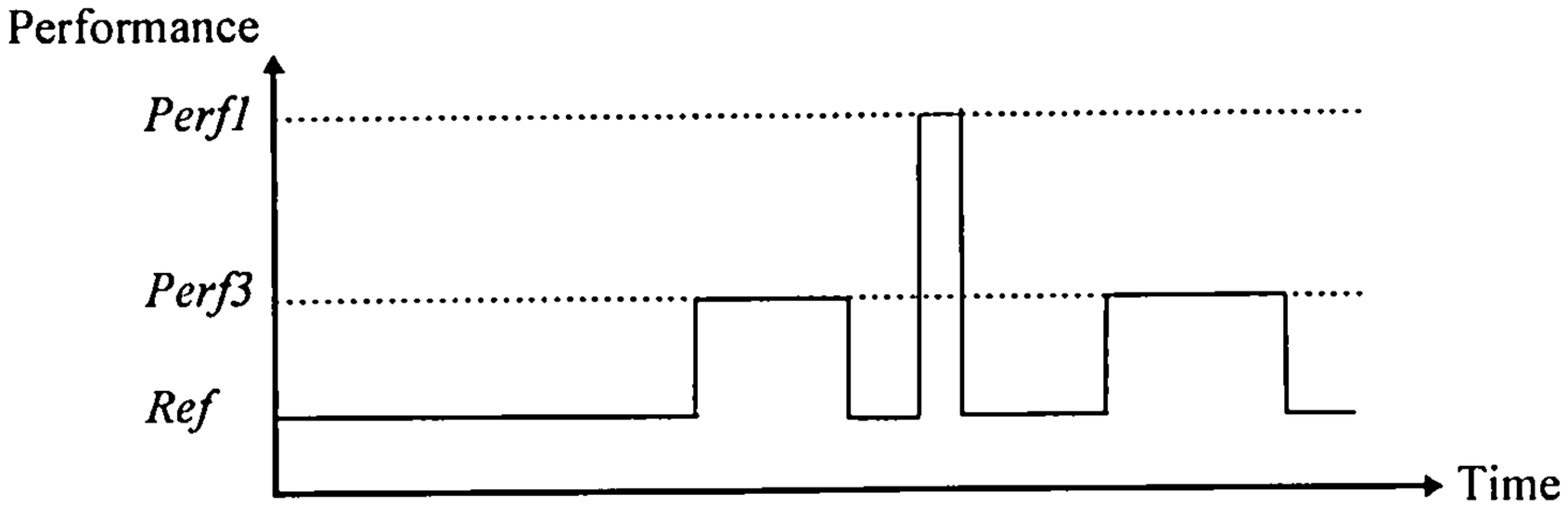


Figure 3.10(b) - Sample path for component 2 for Definition 2

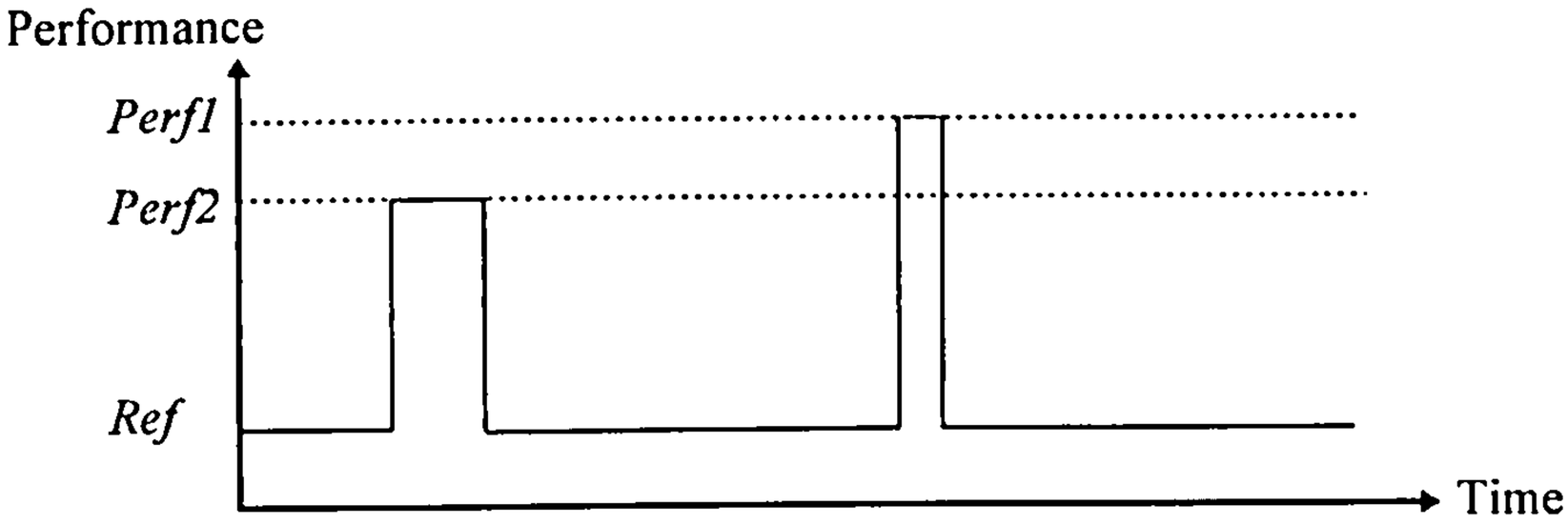


Figure 3.10(c) - Sample path for component 3 for Definition 2

For both definitions, given the different possible combinations of failure rates, mean repair times and performance upon failure, it is likely that a distribution of performability for each component will be somewhat as shown in Figure 3.11.

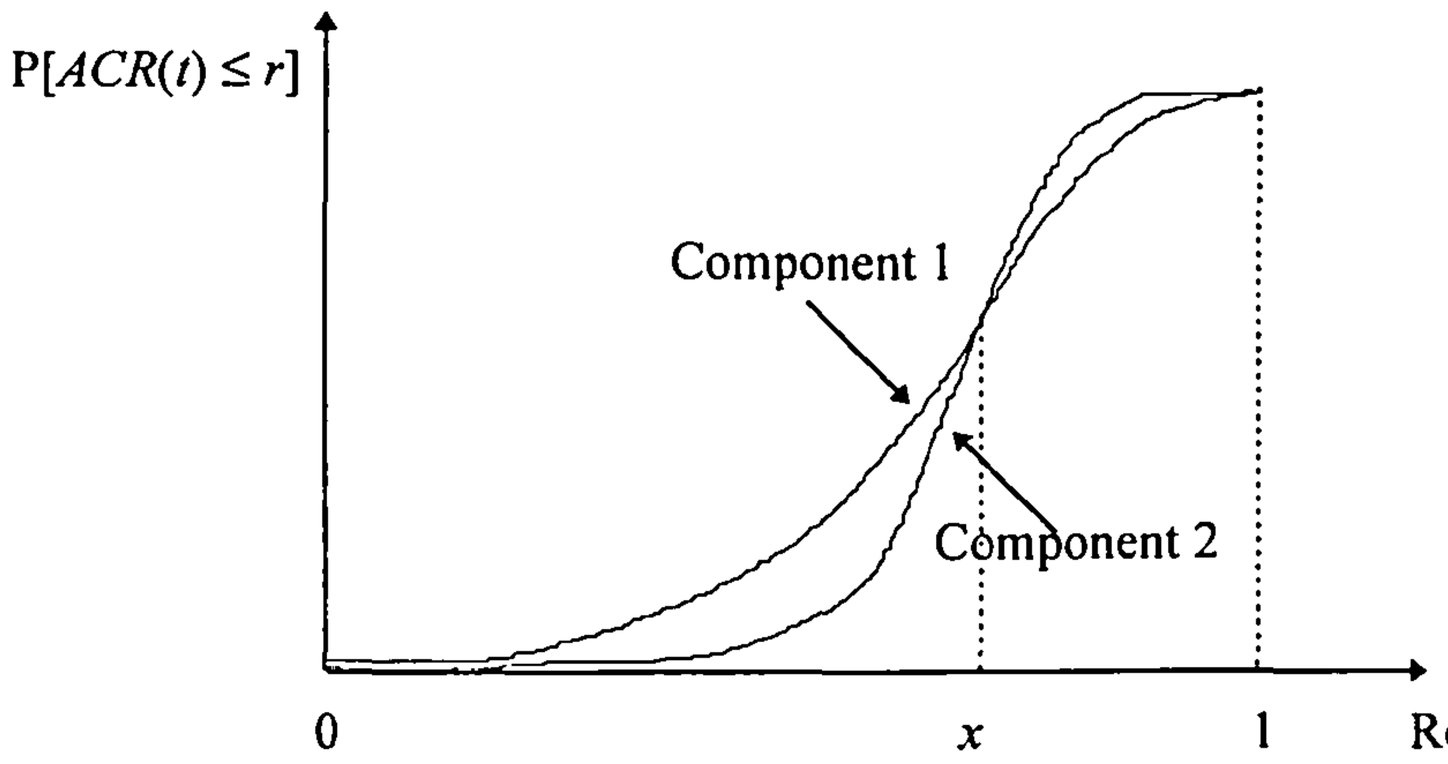


Figure 3.11 - Typical distribution of indices

Therefore, assuming indices are evaluated using definition 1, from rewards in the range 0 to x, component 1 appears to be more important than component 2. At reward equal to x, both components seem to be equivalent and for reward greater than x, component 2 becomes more important. Therefore, the value r_p has to be chosen such that it defines an acceptable average performance level. In a one-to-one mapping of performance on to reward, a given average performance level maps onto one specific accumulated reward and vice versa. For instance, if performance levels vary from 20 to 120, which are mapped directly and proportionally onto rewards 0 to 1, then an average performance level of 100 corresponds to a value of r_p of 0.8. Then, if a span of average performance levels from 100 to 120 is judged to be acceptable, the performability index can be defined as $P[ACR(t) \leq 0.8]$.

3.10.3 Application of the performability index

The following examples illustrate how the performability index can be used. Consider a system of 3 components as shown in Figure 3.12. They could, for example, represent processors P1 to P3, in a distributed system where upon arrival of tasks, a dispatcher decides which processor will execute the task. The following analysis is restricted to evaluation of the importance of each processor and ignores the role of the dispatcher and the arrival process of the tasks.

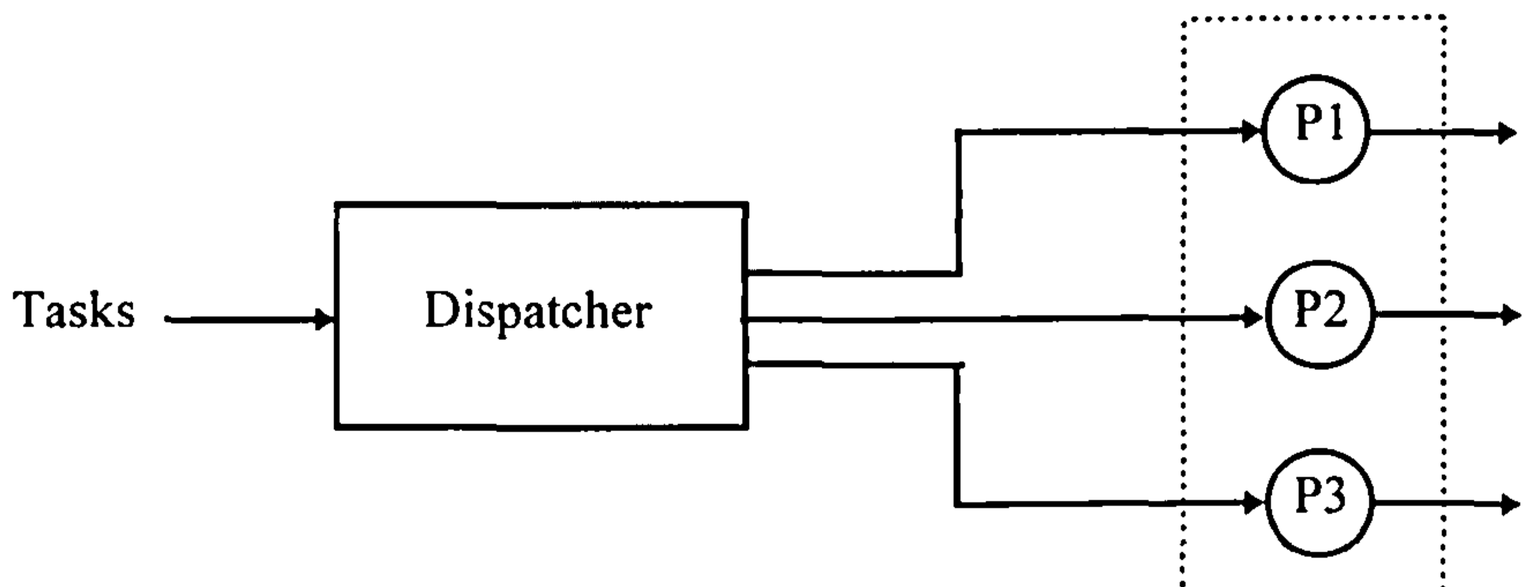


Figure 3.12 - System of three processors

In the first case, assume that the mean repair times of all the processors are equal to 24 hours; and that the three components are associated with failure rates and rewards, which depend on performance upon their failures, as shown in Table 3.1. The better the performance, the higher the value of the reward. The reward associated with the state where all components are functioning is 1.0.

A distribution of performability for all 3 processors assessed on their own, according to Definition 1, is given in Figures 3.13 and 3.14. Over the range of rewards from 0 to 0.7, the processors are ranked in the order P2, P1 and P3 in terms of importance. Between 0.7 and 0.8, there is a switch between P1 and P2 at point of intersection *a* so that the order is now P1, P2 and P3. Another switch between P2 and P3 occurs between 0.8 and 0.9 at point *b*, with order becoming P1, P3 and P2.

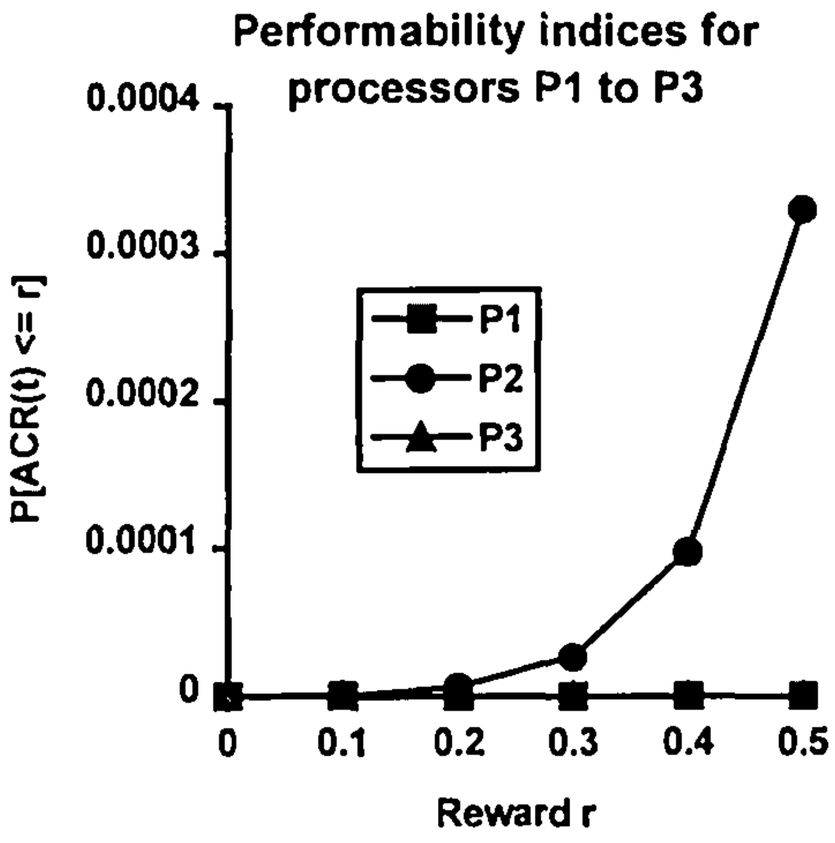


Figure 3.13 - Index in range [0, 0.5] according to Definition 1

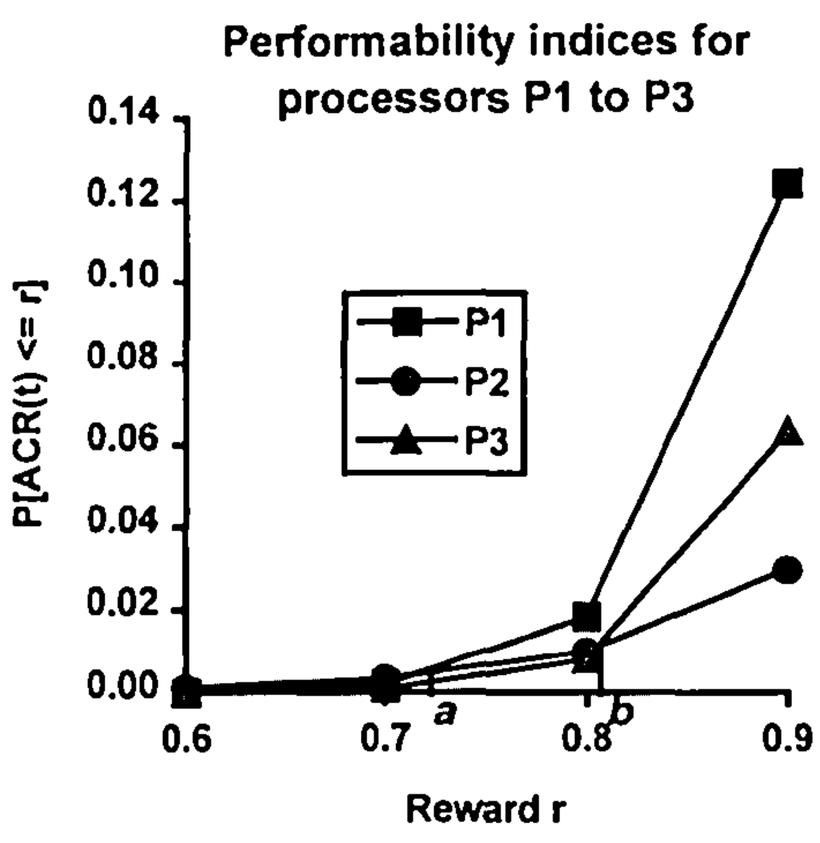


Figure 3.14 - Index in range [0.6, 0.9] according to Definition 1

Let *performability index* be defined as a point performability at reward equal to 0.8.

Component	Failure rate (per day)	Reward	Performability index
P1	0.090	0.5	0.018
P2	0.009	0.0	0.010
P3	0.050	0.5	0.008

Table 3.1 - Performability indices assuming all repair times are equal

Components P1 and P3 are equivalent in terms of degradation caused upon their failures but, given that component P1 fails more often, it is the more important component as shown by its higher *performability index*.

Comparing components P1 and P2, a choice based on performance/reward would give component P2. However, if choice is based on performability, then component P1 is selected. Although component P2 causes more degradation on failure, in terms of performability, component P1 is more important precisely because it fails more often.

Comparing components P2 and P3, a choice based on both performance and performability gives component P2. Even though component P3 fails more frequently than component P2, its performance degradation is not sufficient for it to be considered a more important part of the system being analysed.

In the following example, the same system is considered except that the mean repair times of all the components are not equal. The changes in performability indices are as shown in Table 3.2.

Component	Failure rate (per day)	Mean repair time (hours)	Reward	Performability index
P1	0.090	12	0.5	0.0007
P2	0.009	24	0.0	0.0100
P3	0.050	24	0.5	0.0080

Table 3.2 - Performability indices for different mean repair times

The faster mean repair time of component P1 compensates for its high failure rate so that, as compared with the other components, it is now least important.

Comparing components P1 and P2, both a decision based on performance and performability leads to selection of component P2 which is associated with a lower reward and higher *performability index*. Component P2 does fail less frequently, but a combination of slower repair and lower performance upon failure makes it a more important component.

Comparing components P1 and P3 which are equivalent in terms of performance/ reward, component P3 appears to be more important because, although it fails less often than component P1, it is repaired less quickly.

If performability indices are evaluated according to Definition 2, the distributions are as shown in Figures 3.15 and 3.16, assuming all mean repair times are equal.

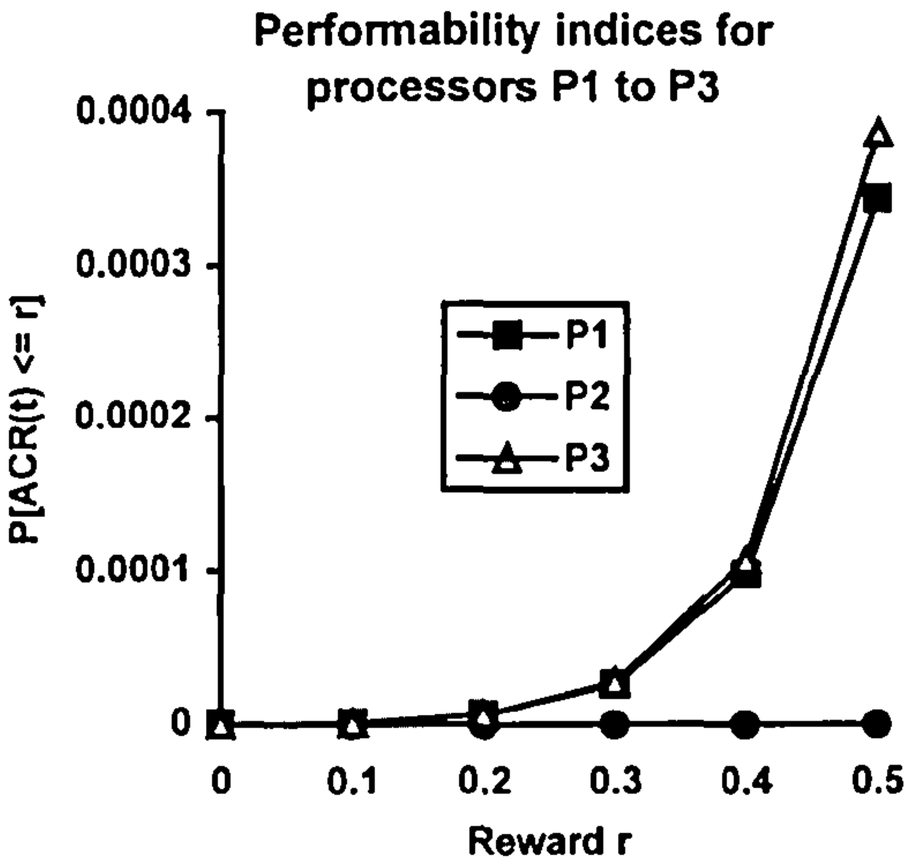


Figure 3.15 - Index in range [0, 0.5] according to Definition 2

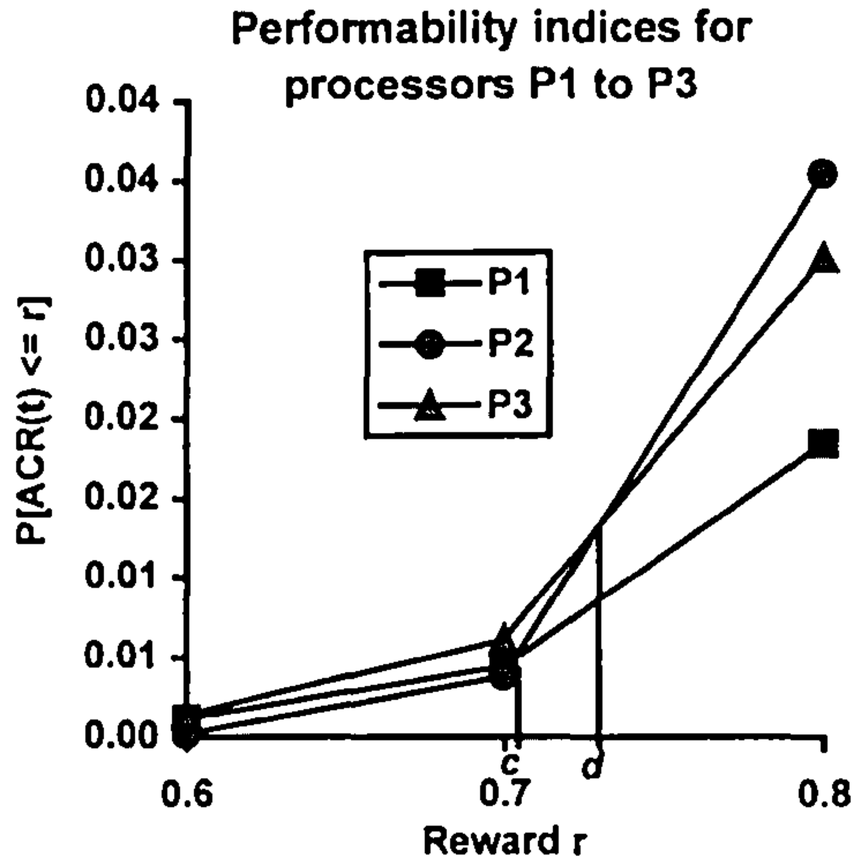


Figure 3.16 - Index in range [0.6, 0.8] according to Definition 2

In the range of rewards from 0 to 0.7, the processors are ranked P2, P1, and P3 in order of decreasing importance. Between 0.7 and 0.8, there are two switches at points of intersection *c* and *d*. For reward between *c* and *d*, the order changes to P1, P2, and P3. At reward equals *d*, another switch takes place with the order becoming P1, P3, and P2. In Figure 3.16, the range shown is restricted to [0.6, 0.8] in order to show the points of intersection more distinctly. From point *d* onwards, the order of importance remains the same until reward is equal to 1.0.

The distributions of performability indices follow the same trend for both definitions but the points at which switches occur are different. This is due to the different combinations of colours, or rewards, which are considered in evaluation of the index. Definition 2 is expected to be more accurate than Definition 1, given that it takes into account the performance associated with failure of the other components; a larger number of combinations of colours, which are obtained if the performability of the whole system is evaluated, are included. However, with this second definition, evaluation of each performability index involves a much higher number of computations as compared with the first definition. This becomes even worse as the size of the system under analysis increases.

Suppose the system were to be equipped with an additional processor which would serve as backup on failure of one of the processors, and assume that this processor has a failure rate of 0.01/day. For instance, if it were connected to processor P1, then in the case where P1 fails, it would take over the job. This processor is used only when processor P1 fails so that it does not affect the performance in any of the other states. Assume all mean repair times are equal to 24 hours. Furthermore, assume that delays involved in this switch are negligible so that any performance degradation during that time is hardly significant. The question is where to add this redundant component. If the performability index is calculated at r_p equals 0.8, then according to definition 1, it would be most beneficial to couple processor P1 with the redundant processor. The least improvement is obtained if it is associated with processor P3. However, based on definition 2, the processors are ranked P1, P3, and P2 in order of decreasing importance so that most benefit is derived from attaching the redundant processor to processor P1, but it is least beneficial to connect it to processor P2.

Performability evaluation for the different cases where the redundant processor is attached to each of the processors in turn gives the results shown in Figures 3.17 and 3.18. In Figure 3.18, only the range of rewards from 0.6 to 0.8 is shown so that the points of intersection are more clearly distinguished. The trend from point f to 0.8 persists up to reward equals 1.0.

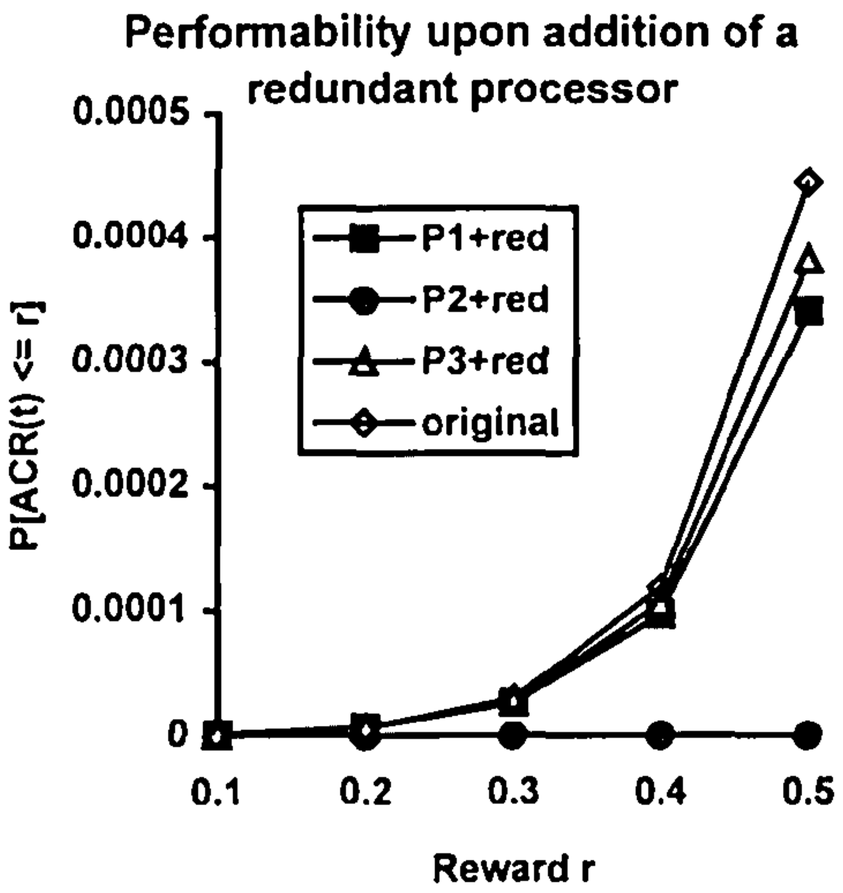


Figure 3.17 - Performability distribution in range [0, 0.5]

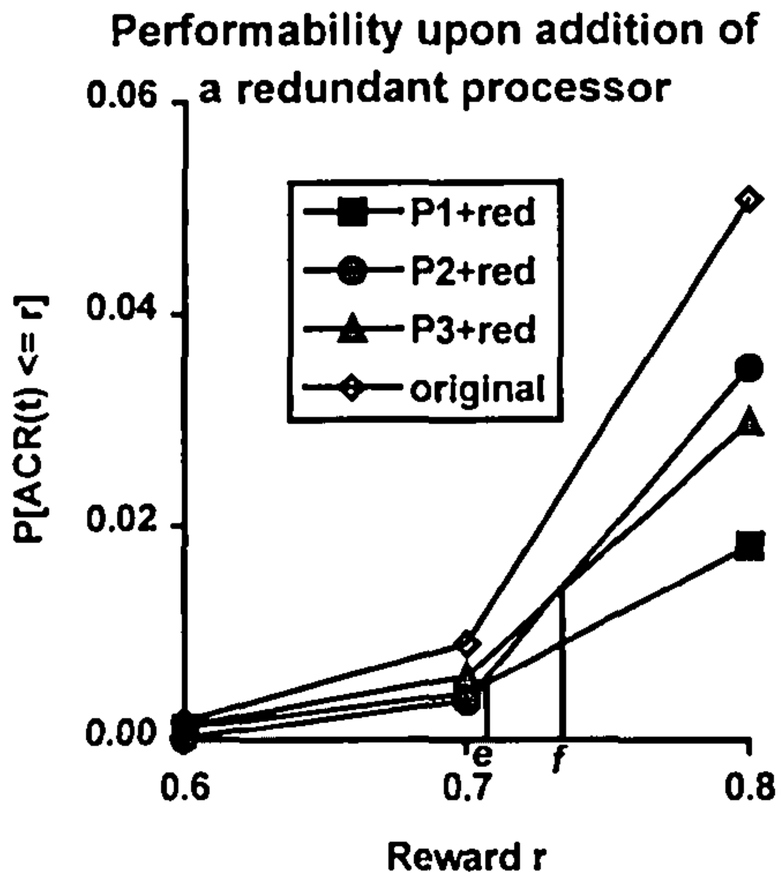


Figure 3.18 -Performability distribution in range [0.6, 0.8]

From reward equals 0 to 0.7, the improvement obtained follows the order P2, P1 and P3 in the sense that most benefit is achieved upon coupling processor P2 with a redundant one, followed by P1 and then P3. At reward equal to 0.9, the order changes to P1, P2 and P3. These results are in accordance with the trend obtained for the performability indices according to both definitions. When definition 1 is used, for rewards between 0.7 and 0.9, the performability index distributions for the various processors intersect at two points denoted by *a* and *b* as shown in Figure 3.14. The distributions obtained upon inclusion of the redundant processor follow the same tendency but the points of intersection *e* and *f* do not exactly correspond to points *a* and *b*. However, when Definition 2 is used, the points of intersection *c* and *d*, shown in Figure 3.16 are closer to *e* and *f*. At reward equals 0.8, the order in which the processors are, based on the improvement derived from coupling each with the redundant processor, matches the order predicted by the performability indices evaluated according to definition 2. The performability index thus gives an indication of the relative importance of each component in the system.

In the example used, the performability distribution of the system, on the addition of a redundant processor for each existing processor, follows the same trend as the distribution of performability for the processor on its own. However, when comparing for all three

processors, the points of intersection are not the same. The performability index, according to Definition 1, fails to be accurate in the regions of intersection; Definition 2 gives better results in these areas.

In this section, the *performability index* has been proposed as a concept, that can be used to resolve those situations where combinations of failure rates, mean repair times and performance degradation upon failure lead to components having different relative levels of importance in a system; two definitions have been proposed. A simple example has shown that although it gives a fair indication of the significance of each component and can be applied generally to determine where to augment a system, Definition 1 is not very accurate in the regions where components are almost equivalent in importance. This is because in evaluation of the index according to that definition, a number of combinations of rewards are completely ignored. Definition 2 gives a more accurate assessment of the importance of components but its evaluation is more computationally intensive because it involves the whole state-space as compared with only two states in the case of the first definition. The underlying assumption when using the performability index to predict with which component the redundant one needs to be coupled is that upon doing so, it does not affect the performance of the system when other components fail. If it does, the performability index cannot give an accurate idea of where to apply this system augmentation.

3.11 Problems associated with Weisberg's result

In some of the test cases considered, the distribution obtained was found to be non-monotonic, which is not acceptable since a probability distribution function should be a monotonically increasing function. The problem was identified as being associated with the evaluation of the term $P[ACR(t) \leq r \mid n, k]$ when using Weisberg's result [WEI71], which involves high-order differentiation. Recall that

$$P[ACR(t) \leq r \mid n, k] = 1 - \sum_{i=1}^{m} \frac{g_i^{(k_i-1)}(\rho_i)}{(k_i-1)!}$$

where $g_i^{(k)}$ is the k^{th} derivative of the function

$$g_i(x) = \frac{(x-r)^n}{\prod_{l=1, l\neq i}^{K+1} (x-\rho_l)^{k_l}}$$

The derivatives of the function $g_i^{(k)}$ can be evaluated recursively using Leibnitz' rule for the k^{th} derivative of a product which gives

$$g_i^{(k)}(x) = \sum_{l=0}^{k-1} \binom{k-1}{0} g_i^{(l)}(x) h_i^{k-1-l}(x) \tag{3.13}$$

where $g_i^{(k)}$ is as defined above and

$$h_i^l(x) = (-1)^l l! \left[\frac{n}{(x-r)^{l+1}} - \sum_{s=1, s\neq i}^{K+1} \frac{k_s}{(x-\rho_s)^{l+1}} \right] \tag{3.14}$$

As k increases, calculation of the individual terms of the summation given by (3.13) involves the manipulation of very large numbers. Any small errors which occur are propagated forward due to the recursive nature of the evaluation of the expression. These errors can cause the result, which is a probability, to be outside the range 0 to 1. If the error is negligible, the result can be assumed to be equal to 1 or 0 depending on the specific case. However, in some cases, the discrepancy is considerable and worsens as the value of n and the required derivative increase. These results contribute directly to the calculation of the inner sum in equation (3.5). By not being in range, they lead to irregularities in the expected trend of results, in the form of non-monotonicity of the distribution function. The above problem was encountered even though a technique [BAR92] which reduces the manipulation of very large numbers was used.

Further investigation of the above problem in both C and Mathematica have shown that the errors are due to accuracy. By increasing the precision of numbers in Mathematica, it is possible to eliminate these discrepancies. In order to cope with cases where accuracy is not high enough, the following check, which is illustrated for three rewards, but which is applicable to a more general case, was imposed to remove any errors which occur during evaluation of the conditional probability.

Given n and r let

$$P[ACR(t) \leq r \mid n, [k1\ k2\ k3]] = a$$

$$P[ACR(t) \leq r \mid n, [k1+1\ k2\ k3-1]] = b,$$

then

$$b < a.$$

A proof of the check is given below.

$$ACR(t)|n,\boldsymbol{k} = \frac{1}{t}\sum_{l=1}^{3}\rho_l\zeta_l \,.$$

For $\boldsymbol{k} = [k1\ k2\ k3]$, let $ACR(t) = \rho_1\zeta_1 + \rho_2\zeta_2 + \rho_3\zeta_3 = x$.

Using probabilistic arguments, the transition times in the case where $\boldsymbol{k} = [k1\ k2\ k3]$ and the case where $\boldsymbol{k} = [k1+1\ k2\ k3-1]$ can be assumed to be the same.

Therefore, for $\boldsymbol{k} = [k1+1\ k2\ k3-1]$,

$$ACR(t) = \rho_1(\zeta_1 + \varepsilon) + \rho_2\zeta_2 + \rho_3(\zeta_3 - \varepsilon) = y\,,$$

where ε is the length of the interval which is now associated with a reward of ρ_1 instead of ρ_3.

Given $\rho_3 < \rho_1$,

$\Rightarrow y > x,$

$\Rightarrow b < a.$

3.12 Problems associated with the many-to-one mapping from performance to reward

As already mentioned, the computation involved in the *randomisation technique* depends heavily on the number of rewards that are allowed. If there are $K+1$ rewards, at n transitions, the number of possible ways of assigning the rewards to the intervals is given by $\binom{K+n+1}{n+1}$, resulting in many possible $\boldsymbol{k}$ vectors which need to be considered in evaluation

of both terms constituting the inner sum of equation (3.5). In order to ease the computation involved, the number of rewards to be considered is restricted to three in the generation of results in this thesis. Therefore, a range of performance levels are mapped on to a reward.

In the consideration of different sets of data, the following problem was encountered. Due to the many-to-one mapping of performance on to reward, performance in some failure states, although degraded, may be judged to be good enough so that the states are associated with the same reward as the fully operational state, where the reward is the highest. Under such conditions, in some of the test cases considered, increasing the failure rate of a component, whose failure results in such a level of performance, causes the distribution of average accumulated reward to be closer to the ideal one. This is because increasing the failure rate of such a component means that the system has a higher probability of being in a state where reward is the highest. This situation is also encountered when the mean repair time of such a component is increased because increasing the mean repair time implies that the residence time in a state where reward is the highest is longer. Therefore, the system appears to be performing better when either the failure rates, or the mean repair times of certain components are increased. Such results are in contradiction with the statement that a distribution of performability gives an indication of the fault-tolerance of a system, where an increase in failure rates, or in mean repair times, should decrease the fault-tolerance capability of a system.

In order to obviate this problem, in comparison of systems with components having different rates of failure and mean repair times, it is essential that performance in the states of failure is distinguished from performance in the fully operational state. Different rewards have to be used to ensure that a state of failure is considered as one where the system is degraded.

3.13 Problems associated with the randomisation technique

A source of error when using the randomisation technique is the fact that, in practice, when conditioning on the number of transitions, n, which occur in a period of length t, the distribution cannot be summed over $n = 0$ to ∞. Instead, it has to be summed over $n = 0$ to

N. But Λ and t determine the number of transitions N for which the performability equation (3.5) needs to be evaluated and N also depends on the pre-specified error tolerance.

Firstly, if t and Λ are very large, special techniques have to be used to compute the Poisson terms in the distribution to avoid overflow or underflow. Secondly, large values of t and Λ mean that N is quite high. As N increases, the number of computations involved increases exponentially. To avoid extremely long running times, optimisation techniques require a considerable amount of memory, of the order of $(N^2x)/2$, where x is the number of states in the Markov chain. Either way, the randomisation technique is not very efficient when Λ and t are large, or the system model consists of a large number of states.

3.14 Techniques of state-space reduction

Depending on the degree of detail included in the model, a performability model may include millions of states. The state-space of a system consisting of c components, each with s different states can grow as fast as s^c. This type of order of complexity constitutes a major source of difficulty in construction and numerical evaluation of detailed and comprehensive models of the behaviour of these systems. One means of dealing with this problem is through state-space reduction techniques such as truncation of the state-space or aggregation of states. The main issue which arises on application of these techniques is how to bound the final error.

Techniques of decomposition and basic aggregation/ disaggregation as used by Courtois [COU77] can be applied when doing a steady-state analysis of performability. Courtois and Semal derived a technique by which different types of bounds on the equilibrium conditional probability of a subset of states in a large and complex Markov model can be obtained with various degrees of accuracy and computational complexity [COU86]. These bounds are computed for certain critical subsets of states, given that the system operates in these subsets, using minimum information on the behaviour of the system in other states.

Muntz *et al.* [MUN89] have extended the basic technique of Courtois and Semal to obtain bounds on steady-state availability. The essence of the method is to generate and solve for only a small subset of states. Given a Markov chain which represents the whole system, each state corresponds to the system state in which some components have failed and others are operational. Real systems are designed to have a high level of availability, so that it is expected that most of the time, the system operates with the majority of its components operational. Based on this observation, most of the probability mass is concentrated on a relatively small number of states in comparison to the total number of states in the model. During its lifetime, the system spends most of its time in this relatively small subset and very rarely reaches the other states of the system. The approach is to maintain a detailed description of the system model for those states where few components have failed, and aggregate the remaining states. The aggregates and transitions between aggregates can be chosen such that upper and lower bounds for availability can be obtained.

As far as transient analysis is concerned, truncation of the state-space generates a simple but not necessarily good bound. In essence, only certain states in the Markovian model are generated and the remaining states are aggregated into a single absorbing state [DES92]. Assume that the rewards, r_i, are bounded, $r_{lb} \le r_i \le r_{ub}$ for all i. Then, the distribution of cumulative reward in the original model is bounded by

$$P^{lb}[CR(t) > y] \le P[CR(t) > y] \le P^{ub}[CR(t) > y],$$

where the superscript lb (ub) indicates that the absorbing state in the truncated model is assigned reward r_{lb} (r_{ub}). The quality of the bounds depends on the probability of reaching a non-generated state during the observation period. Generation of states so as to minimise this probability is an important issue.

A number of techniques have been proposed as means of generating the most probable states [DES92]. Muntz *et al.* [MUN89], through the bounding methodology, assume that a natural partitioning of the state-space exists, such that the majority of the probability mass is concentrated in the states of the initial partitions. This is not an easy task in general, although it works well for availability models.

Li and Silvester [LI84] proposed an ORDER algorithm, applicable to networks, which enumerates m network states s_k, $k = 1, 2, \ldots, m$ in order of decreasing probability and then averages the network performance over these m most probable states. The value of m needs to be predetermined. Yang and Kubat [YAN89] derived a more efficient algorithm that can generate the most probable states s_k in order of decreasing probability until a suitable stopping criterion is met. It does not require prior knowledge of the number of states to be generated and is faster.

Grassman [GRA90] [DES92] implements the randomisation technique in a dynamic fashion. A set of active states is generated at each step, n, of the uniformisation procedure, where an active state is one which is associated with a non-zero probability. This set is adjusted at each step and the transition matrix is generated incrementally according to the current set of active states. Therefore, there is no need to generate the entire transition matrix before the algorithm starts.

Dimitrijevic and Chen [DIM89] [DES92] proposed a dynamic state exploration technique for general Markov models. The generated states are grouped into two subsets, the explored states and the unexplored ones. At each step of the algorithm, the number of visits to each state between visits to the initial state are calculated. Results determine which state needs to be included in the explored and unexplored subsets. The algorithm terminates when a specified stopping criterion is met.

3.15 Clustering or lumping of states of the Markov chain

Lumping is another technique used to reduce the number of states in the Markov chain which should have a direct impact on the amount of memory required during performability evaluation. This is the approach adopted in this thesis to ease computation via a reduction in state-space in generation of results. It is, therefore, described in more detail. Ideally, for the number of rewards being considered, the fully lumped Markov chain would consist of four states: the fully operational state and one state for each of the three possible rewards. However, in order to be able to use the performability equation for the lumped chain, it is necessary that the chain still retains the Markov property and that sufficient detail is

preserved in the lumped model for measures of interest to be evaluated. Also, if solutions obtained for the lumped chain are expected to be the same as those obtained for the original Markov chain, the lumped chain must contain exactly the same information as the unlumped one. In this respect, lumping is subject to certain constraints identified by Kemeny and Snell. A discussion of these constraints is given below as explained in [KEM60].

Consider a Markov chain of s states governed by transition matrix **P** and initial probability distribution π. Let $A = \{A_1, A_2, \ldots, A_r\}$ be a partition of the set of states.

Definition :- A Markov chain is lumpable with respect to a partition $A = \{A_1, A_2, \ldots, A_r\}$ if for every starting vector π, the lumped process is a Markov chain and the transition probabilities do not depend on the choice of π.

Let $p_{iA_j} = \sum_{k \in A_j} p_{ik}$

Then p_{iA_j} represents the probability of moving from state s_i into set A_j into one step for the original Markov chain.

Theorem :- A necessary and sufficient condition for a Markov chain to be lumpable with respect to a partition $A = \{A_1, A_2, \ldots, A_s\}$ is that for every pair of sets A_i and A_j, p_{kA_j} has the same value for every s_k in A_i. These common values $\{p_{ij}\}$ form the transition matrix for the lumped chain.

In lumping of the states of the simplified single failure/ total repair Markov reward model of a network, the fully operational state is left on its own. Lumping is applied only to the failure states where based on the above theorem, the following conditions have to be met. Two failure states can be clustered together if and only if

1. They are associated with the same rewards and
2. Their repair rates are equal, that is there is equal probability of a transition from any of the two states to the fully operational state.

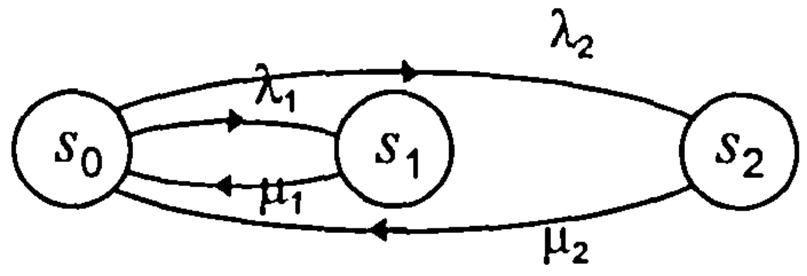


Figure 3.19 - Markov chain of three states

Therefore, given a Markov chain of three states, as shown above, with transition matrix **P**, states s_1 and s_2 can be lumped if they satisfy the following conditions.

1. $\rho(1) = \rho(2)$ and
2. $\mu_1 = \mu_2$.

$$\mathbf{P} = \begin{bmatrix} 1 - (\lambda_1 + \lambda_2)/\Lambda & \lambda_1/\Lambda & \lambda_2/\Lambda \\ \mu_1/\Lambda & 1 - \mu_1/\Lambda & 0 \\ \mu_2/\Lambda & 0 & 1 - \mu_2/\Lambda \end{bmatrix}$$

Then, the transition matrix for the lumped process is given by **P'**.

$$\mathbf{P'} = \begin{bmatrix} 1 - (\lambda_1 + \lambda_2)/\Lambda & \lambda_1 + \lambda_2/\Lambda \\ \mu_1/\Lambda & 1 - \mu_1/\Lambda \end{bmatrix}$$

In the case of the single failure/ total repair model, the transition matrix for the lumped chain is derived fairly easily. If states s_i and s_j are lumped, the probability of a transition from the fully operational state s_0 to the combined state is $(\lambda_i + \lambda_j)/\Lambda$

where λ_i/Λ is the probability of a transition from state s_0 to state s_i,

and λ_j/Λ is the probability of a transition from state s_0 to state s_j.

Then, the probability of a transition from the combined state to state s_0 is either μ_i/Λ or μ_j/Λ

where μ_i/Λ is the probability of a transition from state s_i to state s_0,

and μ_j / Λ is the probability of a transition from state s_j to state s_0.

However, the transition matrix for a general lumped process can be generated in the following manner [KEM60].

Assume that the initial Markov chain is lumpable with respect to a partition $A = \{A_1, A_2, ..., A_s\}$ and that the original chain has r states, whereas the lumped one consists of s states. Let **U** be the s x r matrix whose i^{th} row is the probability vector having equal components for states in A_i and 0 for the remaining states. Let **V** be the r x s matrix with the j^{th} column a vector with 1's in the components corresponding to states in A_j and 0's otherwise. Then the lumped transition matrix is given by

$$\mathbf{P'} = \mathbf{UPV} \tag{3.15}$$

Application of the lumping technique has made possible considerable reduction of the state-space. For instance, in a test case for a system of 40 components, the corresponding Markov chain, which consists of 41 states, was lumped to generate a chain of 6 states; and a second Markov chain of 51 states was reduced to one of 7 states. Given that the lumped model contains exactly the same information as the original one, the same solutions are obtained when carrying out performability evaluation. Therefore, as long as the lumping conditions can be met, there is an advantage to be gained from the technique.

3.16 Evaluating moments of performability [IYE86]

Using the randomisation technique, a distribution of performability has been obtained, which has been interpreted as an indication of the relative fault-tolerance of a system. Although a distribution fully describes the behaviour of the system, moments of performability are measures which may be used to characterise the system. The algorithm to be described is by Iyer *et al.*, where a recursion relating the $n+1^{th}$ moment to the n^{th} moment has been obtained. The first moment corresponds to the mean accumulated reward over a period of time. The *randomisation technique* can also be used to derive the mean

accumulated reward; however, the attraction of the algorithm to be described is that given the n^{th} moment, the $n+1^{th}$ moment can be obtained using recursion.

In essence, the algorithm is as such: an N configuration fault-tolerant system is defined by the process $\{X_s;\ s \geq 0\}$, $X_s \in \{0, 1, ..., N\}$, where X_s describes the configuration (state) of the system at time s. System failures and repairs cause system reconfiguration. In a network context, the system corresponds to a network consisting of N links; each state represents the network in a particular configuration as dictated by failures of the links, only single failures being allowed. A reward function $f(i)$ or f_i is associated with configuration i, where the reward is an indication of performance, and τ_i is defined as the time spent by the system in configuration i over the mission time t.

$$\sum_{i=0}^{N} \tau_i = t \tag{3.16}$$

The average time between the system going into state i and changing configuration as a result of a fault is generally distributed with mean $1/c_i$. It is further assumed that a failure or repair action always causes a system in state i to reconfigure itself into state j ($j \neq i$) with probability p_{ij}.

$$\sum_{j=0}^{N} p_{ij} = 1 \tag{3.17}$$

Then $\{X_s\ ;\ s \geq 0\}$ is a semi-Markov process with state-space {0, 1, 2, ..., N}. The performability of the system over the mission time t is the probability density function of

$$Y_t = \sum_{i=0}^{N} f_i \tau_i = \int_{s=0}^{t} f(X_s)\, ds \tag{3.18}$$

If the times between reconfigurations are exponentially distributed, $\{X_s\ ;\ s \geq 0\}$ is a Markov process with a Markov generator matrix **Q**.

$$\mathbf{Q} = \mathbf{C}(\mathbf{P} - \mathbf{I})$$

where **C** is a diagonal matrix with entries c_i,

I is the identity matrix and

P is a matrix of transition probabilities with entries p_{ij}.

Let **F** be a diagonal matrix with entries f_i.

Let $m^n(t)$ be the vector with entries $m_i^n(t)$ where

$$m_i^n(t) = E\left[Y_t^n \mid X_0 = i\right]$$

$$m_i^0(t) = 1$$

Let $\mathbf{P}(t)$ be the matrix of transition probabilities with entries

$$P_{ij}(t) = P\{X_t = j \mid X_0 = i\}$$

From [IYE86], if $\{ X_t , t \geq 0\}$ is a finite state-space, continuous time Markov chain with finite generator matrix **Q** and if **F** is finite, then

$$m^{n+1}(t) = (n+1)\int_0^t \mathbf{P}(t-\tau)\mathbf{F}m^n(\tau)d\tau \qquad (3.19)$$

$$= (n+1)\int_0^t e^{\mathbf{Q}(t-\tau)}\mathbf{F}m^n(\tau)d\tau$$

The moments are calculated in the following way. Let $\lambda_0, \lambda_1, \ldots , \lambda_N$ be the eigenvalues of the generator matrix **Q**. Assuming that the eigenvalues are distinct, there is at most one recurrent class of states and given that the state-space is finite, there is at least one such recurrent class. Therefore, the chain has a unique stationary probability distribution p vector satisfying the balance equations $p'\mathbf{Q} = 0$. In addition to the recurrent states, there may be transient states which have stationary probability zero. Corresponding to the stationary distribution eigenvector p is a zero eigenvalue labelled λ_0. Then $\mathbf{P}(t)$ has the following representation:

$$\mathbf{P}(t) = e^{\mathbf{Q}t} = \sum_{i=0}^{N} Z_i e^{\lambda_i t} \qquad (3.20)$$

If the eigenvalues of **Q** are distinct, then

$$Z_i = \frac{\prod_{l=1}^{N}(\mathbf{Q} - \lambda_l I)}{\prod_{\substack{l=1 \\ l \neq i}}^{N}(\lambda_i - \lambda_l)} \tag{3.21}$$

Then, if $\{ X_s, s \geq 0\}$ is a finite state-space, continuous time Markov chain with finite generator matrix **Q** having distinct eigenvalues ($\lambda_0, \lambda_1, \ldots, \lambda_N$) and if **F** is finite, then there exists vectors $v_n(i,j)$ such that

$$m^n(t) = \sum_{i=0}^{N}\sum_{j=0}^{n} v_n(i,j) e^{\lambda_i t} t^j \tag{3.22}$$

The vectors $v_n(i,j)$ are defined by the following recursion.

$$v_0(i,j) = \begin{cases} \mathbf{e} & i = j = 0 \\ \mathbf{0} & otherwise \end{cases} \tag{3.23}$$

$$v_{n+1}(i,n+1) = Z_i \mathbf{F} v_n(i,n) \qquad 0 \leq i \leq N, \quad n \geq 0 \tag{3.24}$$

$$\begin{aligned} v_{n+1}(i,j) &= (n+1)\sum_{l=0}^{N}\sum_{k=j}^{n} Z_l \mathbf{F} v_n(i,k) c(j,i,l,k) \\ &\qquad 0 \leq j \leq n, \quad 0 \leq i \leq N, \quad n \geq 0 \\ &+ I(j=0)(n+1)\sum_{l \neq 1}^{N}\sum_{k=0}^{n} Z_i \mathbf{F} v_n(l,k) d(i,l,k) \\ &+ I(j \neq 0)((n+1)/j) Z_j \mathbf{F} v_n(i,j-1) \end{aligned} \tag{3.25}$$

where **e** is a vector of ones, **0** is a vector of zeros,

$$I(j=0) = 1 \quad \text{if } j = 0 \qquad = 0 \quad \text{otherwise} \tag{3.26}$$

$$I(j \neq 0) = 1 - I(j = 0), \tag{3.27}$$

$$d(i,l,k) = \frac{k!}{(\lambda_i - \lambda_l)^{k+1}} \tag{3.28}$$

and

$$c(j,i,l,k) = \frac{(-1)k!}{j!(\lambda_l - \lambda_i)^{k+1-j}} \tag{3.29}$$

The above algorithm is not affected by the number of distinct rewards so that a direct one-to-one relationship can exist between performance and rewards. However, an implementation of the algorithm in Matlab shows that it can be severely degraded as the size of the system being considered increases, in a case where mean repair times were almost equal. This led to the difference between the eigenvalues being very small, which contributed to computational errors in evaluation of the Z_i matrices. For this reason, in the rest of the thesis, the use of moments will be restricted to small networks.

3.17 Conclusions

Although traditional system evaluation has been done with respect to performance and reliability separately, individual assessments of these measures are not sufficient to evaluate and characterise the ability of complex systems to perform, especially in the case of performance degradable systems. This calls for a unified performance-reliability measure, which is referred to as *performability*. In this chapter, the concept has been defined and a mathematical framework for its evaluation has been established.

Performability provides the means of representing a system's ability, expressed by probabilities, to achieve different levels of performance during a specified time interval. *Performability* evaluation involves construction and solution of a suitable model for the system under investigation. Most models are based on Markov analysis. A Markov Reward Model is used as a means of modelling. It consists of an underlying Markov chain which represents structural changes affecting the system; and it is further associated with a reward structure which gives an indication of performance as the system evolves through its various states.

A number of numerical techniques exist for determining performability, many of them being based on the use of double Laplace transforms. The method used for generation of results in this thesis is the *randomisation*, or *uniformisation*, technique. It has been described as a means which can be used to calculate various performability measures and more specifically a distribution of average accumulated reward. The variable Y is identified with the average performance of the network over an observation period $(0, t)$, where performance could refer to the average network delay for instance. Given the mapping of performance on to reward, a distribution of average accumulated reward corresponds essentially to a distribution of the average performance of the system under investigation, if the mapping is one-to-one. In cases where the mapping is not one-to-one, a distribution of average accumulated reward gives an indication of the distribution of average performance.

Performability analysis based on the randomisation technique produces positive results in the sense that the variations obtained in the results when parameters of the Markov chain change lead to the establishment that performability is a measure of fault-tolerance. Tests have shown how the probability distribution function of the average accumulated reward changes when either failure or repair rates, that is, reliability parameters are altered, or the performance levels in each state are modified. With performance levels remaining constant, the system's ability to perform is assessed with respect to different rates of structural changes. When failure and repair rates are kept the same, the performance levels in each state are altered so that the system's performability is analysed with respect to different degrees of performance degradation. In both cases, the changes in results with respect to variations in parameters show that *performability* gives an indication of the fault-tolerance of the system being analysed. The results indicate the ability of the system to withstand failures and to perform in the sense that it can still deliver the required service even though the quality of the service might be reduced. On the whole, the randomisation technique provides a tool which can be used as a means of determining fault-tolerance.

In view of these results, the concept of a *performability index* has been proposed as a means of assessing the relative significance of each component of a given system in terms of its contribution to the average performance, or degree of fault-tolerance, of the system over an observation period. Evaluation of the index takes into consideration the failure rate

and the mean repair time associated with each component as well as the degradation in performance that occurs when it fails. Two definitions of the *performability index* have been proposed. The rationale behind having such a measure is that it can give an idea of the weaknesses of a system, which might require the addition of redundant components, or some form of augmentation. A simple example, based on the addition of one redundant processor to support one of a system of three processors, has been used to demonstrate how the concept can be used. The example also shows that the second definition of the index is more accurate than the first, although it involves more computation for its evaluation. It should be noted that the *performability index* is accurate only in a case where addition of a redundant component does not affect the performance in all the states considered.

During the course of using different sets of data to test the *randomisation technique*, a couple of problems have been encountered. One problem is due to the computational errors arising in evaluation of the term $P[ACR(t) \leq r \mid n, k]$ when using Weisberg's results which employ high-order differentiation. A suitable check was proposed to eliminate these errors. The second problem results from a many-to-one mapping of performance on to rewards, where a failure state and the fully operational state are associated with the same rewards. Not distinguishing between the two states leads to results which are in contradiction with the statement that performability gives an indication of the fault-tolerance of a system. Therefore, it is imperative, especially in comparison of a number of systems, that the reward ascribed to a state of failure is different from the one associated to the fully operational state in order to reflect that a state of failure is one where the system is degraded.

Problems associated with the randomisation technique have also been identified where the main issues are the size of the state-space and the values of Λ and t which govern the number of computations involved. Lumping of states has been proposed as a solution to reduce the number of states of the Markov chain. However, lumping can be applied only if certain conditions are met which preserve the Markov property of the lumped chain and all information present in the original chain. In effect, via a reduction in the size of the state-space, lumping or clustering of states reduces the number of computations involved in performability evaluation.

The *randomisation technique* has been used to derive the performability distribution of a given system. Although the distribution fully describes the behaviour of the system, moments of performability are interesting measures to consider. The algorithm described to evaluate moments is due to Iyer *et al.* [IYE86]. It was implemented in Matlab, and results show that it is fairly restrictive in that, as the size of the system increases, and depending on data relating to repair rates, results could be outside expected bounds. Such results can be obtained for relatively small networks, where the *randomisation technique* gave more reasonable results. Therefore, the *randomisation technique* is preferred as means of evaluating the performability of networks in Chapter 4.

On the whole, given the performability evaluator as an application tool to assess the fault-tolerance of systems, it can be applied to evaluate communication networks and be used in the iterations of network design in order to generate more fault-tolerant configurations.

Chapter 4

Proposal for a performability-based design method

4.1 Introduction

The main objective of the research reported on in this thesis is to derive a design methodology for networks so that they possess a considerable degree of fault-tolerance. In other words, the network sought is one that performs well when all its components are functioning, and can also deliver an acceptable, though degraded, level of performance in the presence of failures. On average, over a period of time, the network is expected to provide an acceptable level of performance, taking into consideration the fact that during that lapse of time, components fail and need time to be repaired. The key issues which need to be addressed are how to obtain a measure of the fault-tolerance of the network, which includes aspects of performance and reliability, and how to approach the design problem itself.

As was explained in Chapter 2, conventional design has differentiated between performance and reliability requirements. Expressing reliability in terms of connectivity requires that the network remains connected when the most likely failure scenarios occur. Performance is assessed, assuming that the network is failure-free, so that any optimisation aims at improving performance under these conditions. Such a design may generate a network where the best performance is attained when all components are functioning, but performance in states of failure may be very bad. Fault-tolerant design cannot be done in this way. In the process of designing a network which performs adequately even under conditions of failure, on the one hand, performance has to be optimised in states of failure, and on the other hand, there is a need to take into consideration how often components fail, and how long they take to be repaired. Performance and reliability aspects have to be combined to reflect the network's ability to withstand failures. In Chapter 3, it was established that performability, in the form of a distribution of average accumulated reward, gives a means of assessing the fault-tolerance of a system. The randomisation technique was

used to generate this distribution, which gives an indication of the distribution of average performance. Therefore, a performability-based criterion needs to be used, as opposed to the pure performance criterion applied traditionally in design.

As was explained in Chapter 2, the design process is iterative, and is made very complex by virtue of the number of parameters and constraints which need to be taken into account. A complete design consists of the establishment of the topological configuration, assignment of capacities to the links with respect to flows and derivation of the most appropriate routing algorithms. These have to be done in order to meet performance and reliability specifications within the limits imposed by a budget. Finding a solution which would take all three into consideration simultaneously, and be globally optimal with respect to the constraints, is an almost impossible task. A more feasible approach is to deal with each sub problem, as identified by Gerla and Kleinrock [GER77], separately.

Following this line of thought, a new design strategy is put forward which firstly based on initial ideas from [BAR92], is performability-based and secondly is split into the following three different stages:

1. the *initial topology generation* stage
2. the *connectivity establishment* stage and
3. the *fault-tolerance augmentation* stage.

The first two stages are used to generate the initial minimal network which meets basic reliability requirements. A 2-connectivity criterion holds so that upon any single link failure, the network still remains connected. In the third stage, the question of how to equip the network with sufficient resources for it to be able to withstand failures is addressed. Only link failures are considered. The network is subjected to two fault-tolerance augmentation mechanisms which are improving connectivity and increasing capacity.

Sufficient redundancy needs to be built in to ease flow and maintain performance even under conditions of failure. On the one hand, the addition of links improves the reliability of the network, if reliability is expressed in terms of connectivity; and on the other hand, by providing an alternate path for traffic, this mechanism avoids a drastic degradation in performance on the occurrence of a link failure. The capacity of the links has

to be sufficient not only to deliver the required performance when all components are functioning, but also to be able to accommodate extra diverted traffic in case of breakdowns in the network. Although capacity augmentation does not improve the reliability of the network, it reduces degradation in performance when failures occur. Both mechanisms, therefore, contribute to augmenting the fault-tolerance of the network.

Choice of how and where to apply the above mechanisms is another crucial point. Traditionally, network designers have applied optimisation techniques to the network, assuming that all components are functioning. As a result, resources are placed in such a way as to attain the best performance when the network is fully operational. This does not necessarily mean that this configuration is optimal with respect to achievable performance when a component fails. In this thesis, fault-tolerance augmentation is not applied to a fully operational state, but to a *failure state*, that is, the network to be optimised is in a configuration where a particular component has failed. By optimising in a failure state, it is expected that the network will be better equipped to face a condition of failure, and performance degradation will be reduced, thereby leading to an improvement in fault-tolerance.

Once the state is chosen, a number of possible ways to augment the network can be found. Questions arise as to which is the best. Is it more beneficial to add a new link or augment the capacity of an existing link? Where should a link be added to get the best improvement in fault-tolerance? Traditionally, choice of the best mechanism is based on performance only. The one which gives most improvement in performance in the state being considered is applied. Although Kang and Tan [KAN92] apply capacity augmentation to a failure state, they select the best link with respect to performance in that state itself. In this thesis, the choice of the augmentation mechanism is not based on performance alone. It is based on the concept of *performability*. The mechanism is chosen, which gives the best improvement in performance not only in the state where optimisation is being carried out, but also in all the other configurations in which the network can be. Furthermore, by using a performability-based criterion to select the topology, data related to the failure rate and the mean repair time associated with each component is also being taken into consideration.

Therefore, the network aimed at is one which can deliver a good average performance, over a period of time, taking into account the occurrence of failures and repairs.

Apart from the main stages of the proposed design method, a number of other issues need to be dealt with, namely establishing the cost of the network, and evaluating its performability. Two definitions are proposed to determine cost. The first one represents the cost of the network in terms of its components, and the second definition includes this initial cost as well as the average cost of repair expected to be incurred over a certain time period. Computation of performability requires evaluation of the performance of the network in all the states in which it can be, as dictated by single failures of its links. Choice of a suitable performance measure and an appropriate routing algorithm are important. Although a particular performance measure, and routing algorithm, may be reasonable in providing a means to assess a network in a fully operational state, they may not be adequate when performance is considered in states of failure. Both the routing algorithm and the performance measure have to be such that they reflect degradation when a component fails. Two networks are used to discuss problems related to this issue, thereby leading to a Gradient Projection Method (GPM)-based algorithm [BER87] being adopted for routing purposes in the proposed design technique.

The rest of the chapter is organised as follows. An overall diagrammatic view of the stages of the design and the required inputs is given in Section 4.2.1. The assumptions governing the design method are stated in Section 4.2.2. The stages of the design method are then explained, with the *initial topology generation* stage being considered in Section 4.2.3, and the steps of the *increase in connectivity* stage being outlined in Section 4.2.4. More emphasis is put on a discussion of the *fault-tolerance augmentation* stage in Section 4.2.5, where the steps of this iterative stage are explained. The issue of choosing a state for optimisation is treated in Section 4.2.5.1. A discussion of how to use the concept of the performability index in the process is given in Section 4.2.5.2. Having chosen a state, the augmentation strategy, described in Section 4.2.5.3, needs to be considered. A set of rules, which can assist in deciding on where to add potential links, or increase the capacity of existing links, is proposed in Section 4.2.5.4. Choice of the final topology is discussed in Section 4.2.5.5.

During the course of proceeding through the iterations of the *fault-tolerance augmentation* stage, a performability-based descent factor needs to be used. This is defined in Section 4.3. As explained in Chapter 3, the evaluation of performability requires a prior mapping of performance levels on to rewards. Two definitions for this mapping are given in Section 4.4. The descent factor also includes cost. In Section 4.5, two definitions of cost are proposed. The issues related to choosing the right performance measure, and routing algorithm, are discussed in Section 4.6. Two networks are used to illustrate the problems that may arise. The first example, which addresses the problem of how to assess the performance of a network under conditions of node failures, is discussed in Section 4.6.3. In Section 4.6.4, problems resulting from application of a non-optimal routing algorithm are discussed. This leads in Section 4.6.5 to a description of an optimal routing algorithm, based on a Gradient Projection Method. It is applied to determine the performance of the two networks used previously. This algorithm is adopted as routing algorithm in the proposed design technique. The chapter is concluded in Section 4.7.

4.2 Design Proposal

4.2.1 Synopsis of design proposal

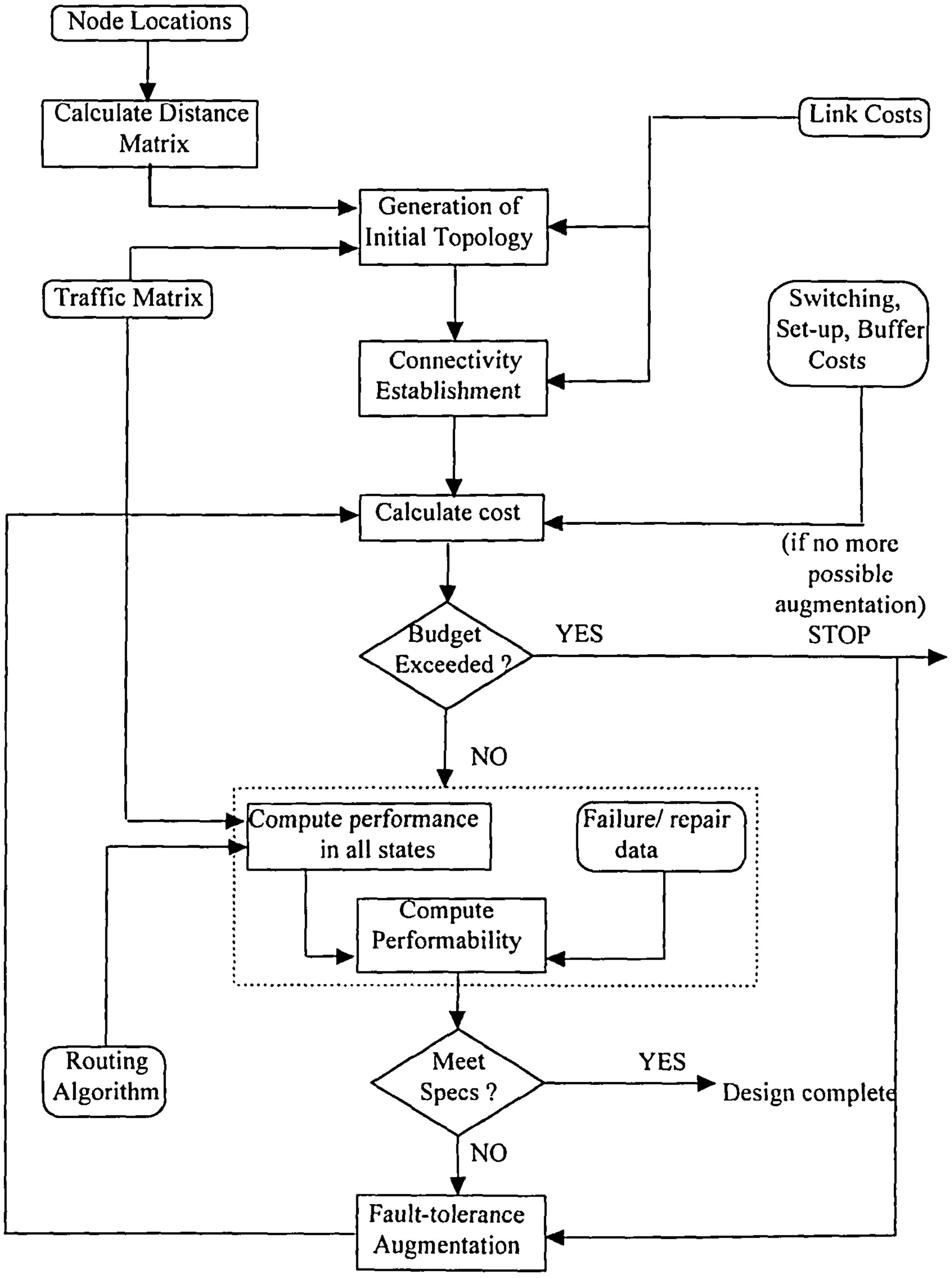


Figure 4.1 - Schematic representation of the overall design method

Although the design proposal itself comprises only three stages, a number of other stages are required, the main ones being establishing the cost of the network and evaluating its performability. The two questions that need to be addressed after each iteration are whether the allocated budget has been exceeded and whether performability specifications have been met. During the *fault-tolerance augmentation* stage, a number of topologies are derived, whose performability and cost need to be evaluated. If the cost of each possible topology exceeds the budget, the design process is stopped. The end topology is the one obtained at the previous iteration. As long as the budget is not exceeded, the iterations of the third stage can proceed until performability requirements are met, or until the best achievable network is designed.

The inputs to the design consist of the node locations, traffic demands and various attributes which contribute to evaluation of the cost of the network. Given the routing algorithm and traffic demands, the performance of the network can be derived in all the failures states which can be considered. Failure and repair data of the different components are required to determine the performability of the network.

For even a small network, the number of states in which it can be, as dictated by failures of its components, can be very large. If the network consists of n links, then 2^n possible configurations can be identified due to the different combinations of failures which can occur. This problem has a serious implication on the computational complexity of the design algorithm. Firstly, cost and performance need to be determined for each of these configurations. Secondly, consideration of all these states results in creation of a huge state space in the performability model. Size of the state space has a direct bearing on the size of the transition matrix and reward structure governing the Markov Reward Model, hence memory requirements. Evaluation of performability also becomes more computationally intensive.

In order to alleviate the consequences of considering all combinations of failures, it is assumed that only single failures can occur. Mean repair times are assumed to be very short as compared with mean time between failures so that once a failure has occurred, the

likelihood of a second failure within the time frame when the component is being repaired is small. Therefore, a network of n links can now be in only n failure states.

4.2.2 Assumptions

Throughout the design, the following assumptions are made.

1. All node locations are known. Design involves connecting them so as to achieve the best level of performability under cost constraints.

2. Links are unidirectional so that bi-directional communication between a pair of nodes is ensured by two links carrying traffic in opposite directions. However, failure of any of these links is treated as failure of both links, that is total breakdown of communication between the connecting nodes.

3. A node consists of switching components and buffers for each of the outgoing links at the node. The queueing discipline governing these buffers is First In First Out.

4. The traffic matrix is known prior to design and remains fixed.

5. The routing algorithm does not change in stages 1 to 3. An optimal routing algorithm based on the Gradient Projection Method [BER87] is used.

4.2.3 Initial topology generation stage

This stage consists of generating the physical network, or transmission network, so as to ensure connectivity of all the nodes. For a given number of nodes, the minimal spanning tree that connects them can be generated. Derivation of this network can be based on distance, or traffic demands. In other words, the cost, or weight, imposed on the links can be the actual physical length of the links, or the traffic demands between the connecting nodes. In a sense, the algorithm used is similar to Kruskal's algorithm with no capacity constraints imposed, as outlined in Section 2.3.8.1.3 of Chapter 2. If generation of the graph is distance-based, then at each iteration, the shortest link, that is the least-cost link is chosen. If the algorithm is traffic-based, then connection is established between the nodes where traffic

demand is highest. Either way, the minimal spanning tree is always the same, irrespective of the starting node.

Once the spanning tree is obtained, the flows on all the links can be determined and an optimal capacity assignment algorithm, the *square root channel capacity assignment* method [KLE76] can be applied to generate values for the capacities of the links. The algorithm was explained in Section 2.4.1 of Chapter 2.

4.2.4 Connectivity establishment stage

Since the starting topology is a minimal spanning tree (MST), failure of any link leads to two disconnected networks. Such a network where there is only one possible path between any pair of nodes is clearly unreliable. At least a 2-connectivity criterion needs to hold for all nodes, that is, there should be at least two distinct paths for traffic to flow between any pair of nodes. In this stage, more links are added to improve connectivity, and satisfy this criterion. Given that the aim is to provide solely improved reliability, the addition of more links to improve connectivity need not be performability-based. The establishment of which links to incorporate into the network can be based on cost, or traffic, alone, depending on the criterion used to generate the minimal spanning tree. Each link added at this stage is of a minimum allowable capacity. The procedure adopted to improve the connectivity of the network is described in the following steps and illustrated in Figures 4.2(a), (b), and (c).

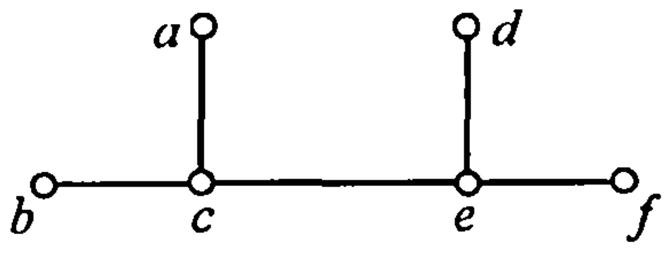


Figure 4.2(a) - Minimal Spanning Tree connecting nodes

Pass One

For each node,

1. If node is connected to MST via a single link, for example node *a* in Figure 4.2(a)

   (i) Consider links from that node to other nodes except the

one it is already connected to.

(ii) Establish cost in each case.

(iii) Add the least-cost link, provided the budget is not exceeded.

If addition of links is based on demands, then a connection is established between the pair of nodes where traffic demand is highest, as long as this does not cause the budget to be exceeded.

In the example used, assume that cost is the criterion on which addition of links is based. Therefore, in Figure 4.2(a), such a node is node *a*, which can be connected to node *b*, assuming this is the shortest link. At the next iteration, node *d* is connected to node *f*, giving Figure 4.2(b).

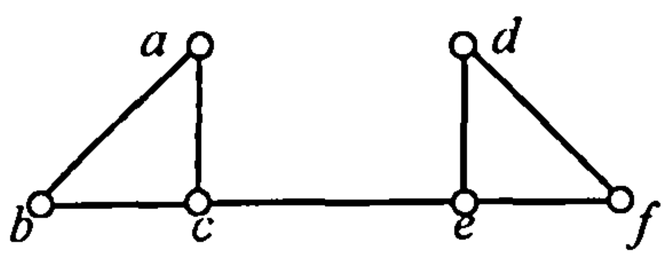


Figure 4.2(b) - Network generated after Pass One of the *increase in connectivity* stage

Pass Two

In some cases, it may happen that the network is still not entirely 2-connected. For instance, in Figure 4.2(b), failure of the link between nodes *c* and *e* causes disconnection of the network. *Pass Two* is required to find out if such conditions exist. The least-cost link, or highest demand link, between other nodes in the disconnected clusters is then added. A different path is thus established between the two clusters. A link can be added between nodes *a* and *d*, giving the network in Figure 4.2(c).

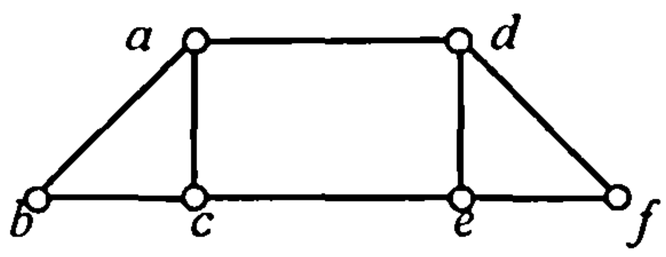


Figure 4.2(c) - Network generated after Pass Two of the *increase in connectivity* stage

It should be noted that the procedure described generates 2-connected networks with respect to link failures only. Therefore, at the end of this stage, there exist two distinct link-disjoint paths between every pair of nodes. It is possible that changes can be brought about to the topology generated in order to obtain a better network in terms of cost. For example, the links connecting nodes *a* and *c*, and nodes *d* and *e*, can be removed without violating the 2-connectivity criterion. In this thesis, any changes to the topology at this stage are not considered. Given that the main objective is to augment the fault-tolerance of the network, the topology obtained at the end of Stage 2 is used directly as starting topology in Stage 3.

However, if changes to this topology were to be investigated, performability can be used as criterion to choose between the various configurations that can be derived. A performability-based criterion can also be used in *Pass Two*. Such a factor is not particularly suitable for *Pass One*, where in most states of failure, the network is disconnected and performance degrades drastically.

After the *connectivity establishment* stage has been completed, the cost of the new topology can be determined, and its performability can be assessed. If the budget has not been exceeded, there is still room for further addition of redundancy and capacity and the *fault-tolerance augmentation* stage can be brought into play.

4.2.5 Fault-tolerance augmentation stage

Fault-tolerance augmentation involves equipping the network with enough resources so as to be able to avoid loss of traffic and large delays when failures occur. Ideally, quality of service must remain unaltered, irrespective of whether all components are operational or not. Hence, a direct implication is that sufficient redundancy coupled with adequate capacity have to be built into the network for it to cope with any excess traffic under conditions of failure. Two fault-tolerance augmentation mechanisms are proposed. They are

(i) adding links, and

(ii) augmenting the capacity of existing links.

This stage is iterative in nature. At the beginning of an iteration, there is a network whose performance can be assessed in all of its failure states; and its performability can thus be evaluated. Any augmentation applied to the network can then be done according to the steps outlined below. Each of these steps is discussed in more detail in the sections following.

At the beginning of an iteration, let performability and cost of the network be $Perf_{old}$ and $Cost_{old}$ respectively.

Step 1. Choose the state where augmentation is to be tried.

Step 2. Identify most heavily loaded links.

Step 3. Depending on the location of most heavily loaded links, apply the rules of augmentation to decide on possible mechanisms.

For each possible augmentation mechanism and resulting topology, compute

$$\frac{Perf_{new} - Perf_{old}}{Cost_{new} - Cost_{old}}, \text{ that is } \frac{\Delta Perf}{\Delta Cost}$$

where $Perf_{new}$ and $Cost_{new}$ are associated with this new topology.

Step 4. Choose the best topology with respect to above ratio, that is, the one giving $\frac{\max \Delta Perf}{\min \Delta Cost}$ and apply augmentation technique.

Step 5. Repeat the steps of iteration until performability specifications are met within the allocated budget.

Note that if the budget is assumed to be unlimited, there is no need to include cost considerations in the descent factor. Furthermore, if the budget is limited, a performability-only based descent factor can be used and the design process is stopped once the amount of money available is used up.

4.2.5.1 Choice of state

Choice of the state where augmentation is to be carried out is an important decision. The state of the network here refers to the possible configurations it can be in, as dictated by the occurrence of failures and repairs. Given that only single link failures are allowed, a

network consisting of n links can be in $n + 1$ states: one state when all links are operational and n states, each of which being one where one link has failed.

Traditionally, designers have applied augmentation to the fully operational state, with no regard for performance in the failure states, as in the *cut saturation* method described in Chapter 2. This is so because design has concentrated on obtaining the optimum network when all components are functioning. The capacity augmentation techniques, described in Chapter 2, do optimise in a state of failure. In the proofing method [NIS88], the capacity and flow assignment problem is solved for each state in which the network can be. Capacities of the links are chosen based on the values allocated in each state, which can be an expensive approach. It is not necessary to optimise in every state in order to find a good combination of capacities which gives adequate performance in all the states. Kang and Tan [KAN92] optimise in a state of failure, but do so with respect to performance in that state only.

In this thesis, the state chosen is a *failure state*. Since the aim of the research is to achieve a fault-tolerant design, it is reasonable to optimise the network in a state of failure. Resources are placed in the best possible way to reduce performance degradation when a link fails. Even if the network is being considered in a failure state, any augmentation applied should contribute to improving the performance in the fully operational state, although the network might not be optimal in that state.

Choice of the state of failure can be based on performance or performability. In other words, the state of worst performance, or the one associated with the worst performability index, can be selected. It makes sense to investigate possibilities of augmentation in the state where most degradation in performance occurs, as in Kang and Tan [KAN92]. However, such an event may be rare and it may be more beneficial, over a period of time, to apply augmentation to a state where less degradation occurs, but failure of the particular link is more frequent. Use of the performability index in choice of state is therefore proposed. The concept was introduced in Chapter 3 and is discussed briefly in Section 4.2.5.2.

4.2.5.2 Concept of performability index

The *performability index* was proposed as a concept which gives an indication of the relative importance of each component in the network by virtue of its failure rate, mean repair time and degree of performance degradation caused by its failure, over a period of time. The rationale behind having a *performability index* is to differentiate between a state of failure where performance degradation is fairly low but component fails frequently, and one where performance degradation is drastic but such an event occurs very rarely, assuming that they have similar repair times. In such a situation, the question is whether it is preferable to apply augmentation to a state of worst degradation/ low failure rate or one of less degradation/ high failure rate. The *performability index* is expected to provide a means of deciding which component is more important. Mean repair times are also taken into consideration in evaluation of the *performability index* so that for instance, it can be used to differentiate between a low performance/ low failure rate/ slow repair state and one of high performance/ high failure rate/ fast repair.

Two definitions of the *performability index* for a component have been proposed in Chapter 3. In the first one, each component is assessed on its own, whereas the second definition considers the system assuming that when the component in question fails, performance degradation is minimal. Although the second definition tends to be more accurate in indicating where any form of augmentation is more favourable, its evaluation demands more computation and is not justified for each component at each iteration of design. Therefore, the first definition of *performability index* is used in generation of results given in Chapter 5.

The performability index is defined as $P[ACR(t) \leq r_p]$, where $ACR(t)$ is the average accumulated reward over an observation period t and r_p is a specified reward level. The index associated with each link is determined. As explained in Chapter 3, assuming a mapping of performance on to reward which is such that the poorer the performance, the lower the reward, the distribution of $ACR(t)$ aimed at is a step function $\mathbf{u}(r - 1.0)$, where the maximum reward is equal to 1.0. Therefore, at a reward of r_p, the higher the value of $P[ACR(t) \leq r_p]$, the more important is the link, whose performability index is being

evaluated. Hence, the state associated with the highest performability index is chosen at that stage.

4.2.5.3 Augmentation strategy

Once the most loaded links have been identified in a particular state, the augmentation strategy consists of either adding links to deviate traffic from most loaded links and nodes, or augmenting capacities of most utilised links. Although both mechanisms can be combined, results have been obtained for cases where design is based on either an increase in connectivity, or a capacity augmentation, in which case the following procedures apply.

Mechanism 1 - increase in connectivity

Redundancy needs to be added in order to relieve most congested links, or bypass highly congested nodes. Having determined the load on all the links, the rules described in Section 4.2.5.4 can be used to identify the scenario being dealt with. In particular, since design is based on an increase in connectivity, rules 1 to 3 apply.

Given a number of links which can be added to the network, choice of the most suitable one is based on the following steps.

For each possible link and resulting topology,

(i) determine cost,

(ii) if maximum cost is exceeded,

consider next link.

(iii) otherwise

determine performability.

The performability-based descent factor is used to decide which link to add to the network. The above steps are repeated until performability specifications are met within the allowed budget.

Mechanism 2 - capacity augmentation

In this case, the question is where to apply capacity augmentation in order to achieve maximum benefit in terms of performability. If the number of links is not large, then augmenting the capacity of each of them can be assessed. Otherwise it can be tried for most loaded links only. Although capacity augmentation can be tried in any scenario, it is found to be particularly suitable when rules 4 and 5 apply.

An increase in capacity of a highly utilised link in a state of failure should relieve congestion and improve performance, thereby increasing the fault-tolerance of the network. Therefore, for each network generated upon a capacity augmentation, the steps outlined in the case of addition of a link are applied to determine performability and cost. The performability-based descent factor is then used to select the best topology. Again, the steps are repeated until the performability objective is attained, or until the budget is exceeded.

4.2.5.4 General rules for fault-tolerance augmentation

Increase in connectivity and capacity augmentation can be combined to generate the best network. Investigation of different possibilities of where to add links in a network and where to apply capacity augmentation has led to a statement of the following rules. Although these rules do not cover all the scenarios which can occur in terms of distribution of load across the network, they can assist in deciding where to apply augmentation.

1. If a node is particularly congested, it is likely that all links into and out of the node are overloaded. Then, the mechanism is to try and deviate traffic away from that node. For example, in Figure 4.3, if all existing links, indicated by solid lines, into and out of node $\mathcal{N}$ are overloaded, then potential links which can be added are shown by the dotted lines, which avoid direction of traffic into node $\mathcal{N}$.

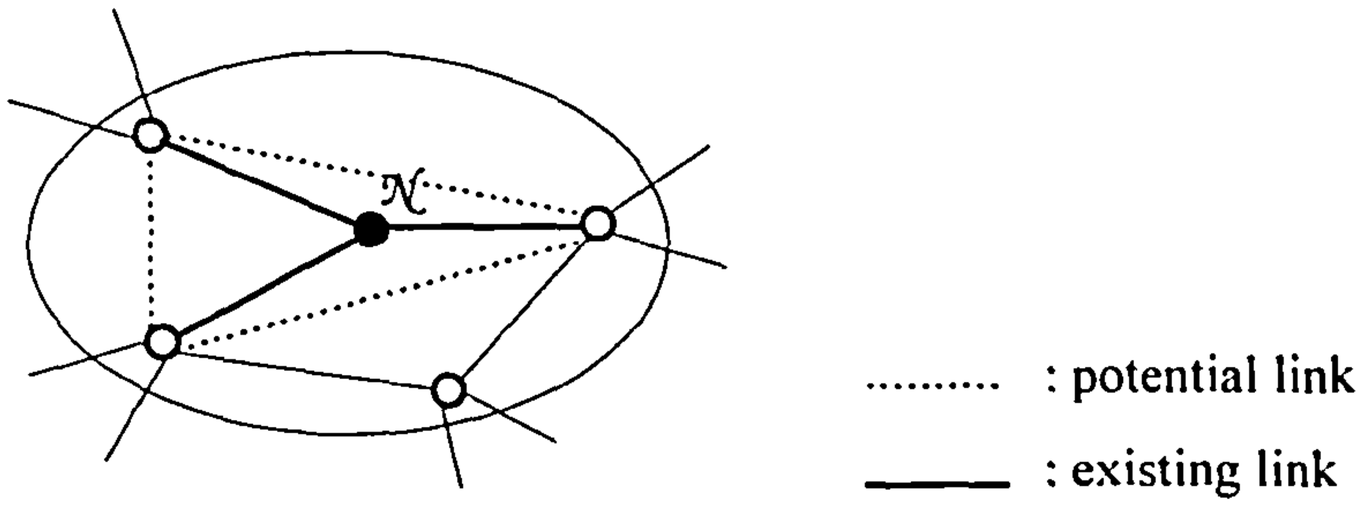


Figure 4.3 - Scenario for application of rule 1

2. If a link is identified as being most loaded, then the node from which it originates is found. The potential augmentation links are links from that node to other nodes in the networks. For example, in Figure 4.4, if link *l* is overloaded in the direction originating from node $\mathcal{N}$, then possible links are the ones from node $\mathcal{N}$ to other nodes in the network, as indicated by the dotted lines.

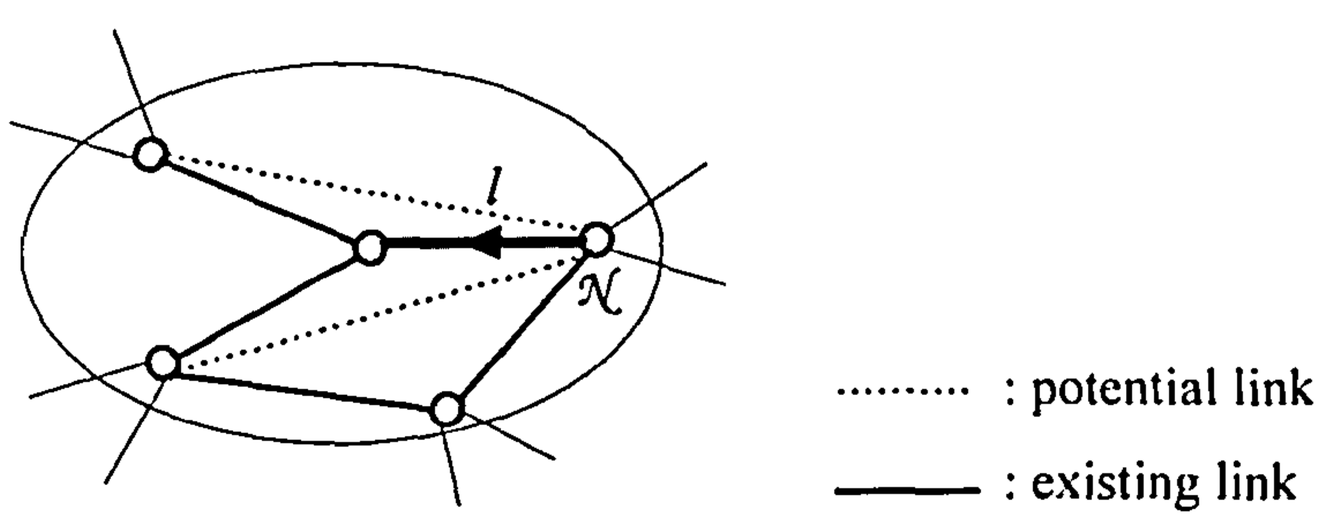


Figure 4.4 - Scenario for application of rule 2

3. If the most loaded link is the sole connection between a particular node and the rest of the network, then other links to be considered are links from that node to other nodes in the network. For instance, in Figure 4.5, even if the most loaded link leads into node $\mathcal{N}$, given that it is the only means of communication between the node and the rest of the network, links to be considered are as shown by the dotted lines.

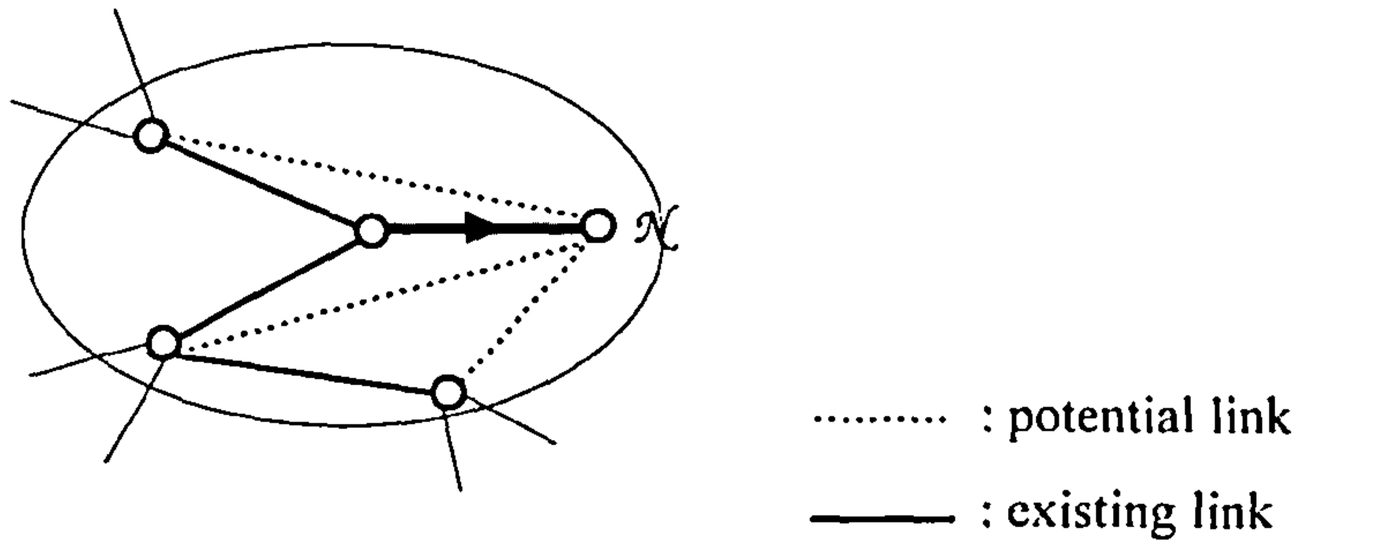


Figure 4.5 - Scenario for application of rule 3

4. If certain links are identified as being particularly overloaded in any situation described in rules 1 to 3, then augmenting their capacity is worth investigating. This can be compared with the addition of potential links as already outlined.

5. If overloaded, or most loaded links, are scattered as shown in Figure 4.6, then the approach is to assess augmentation of the capacity of each of them.

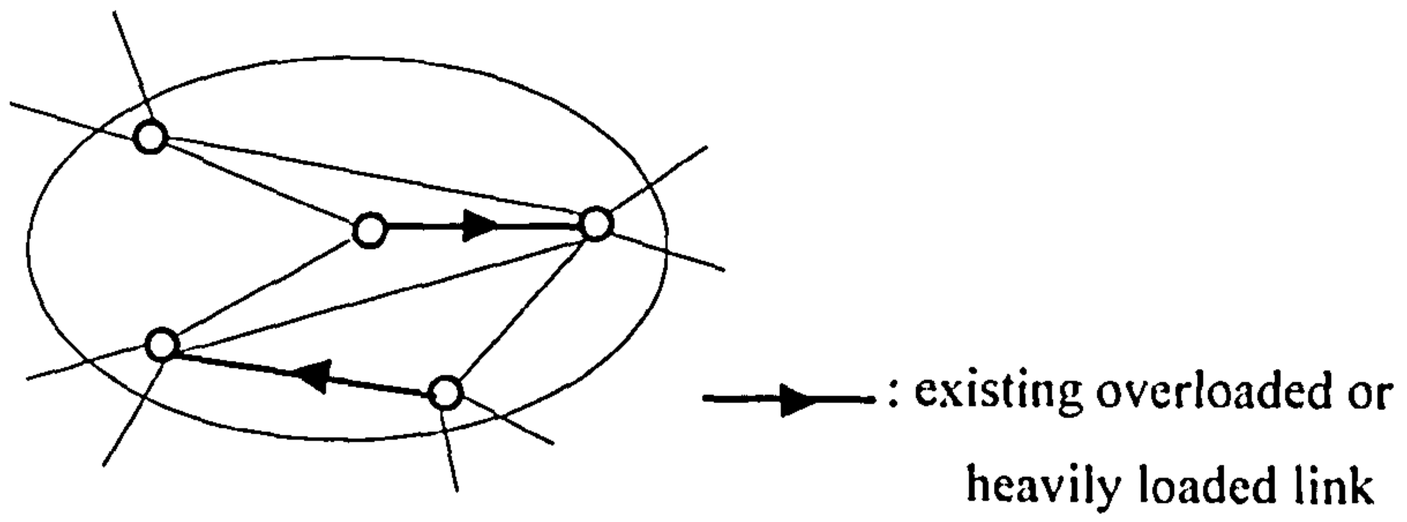


Figure 4.6 - Scenario for application of rule 5

4.2.5.5 Choice of topology

Although the aim is to use performability to select the final topology, different possibilities are investigated for comparison in the generation of the results. The topology chosen at the end of each iteration can be

1) the one performing best in the fully operational state, as is done traditionally,
2) the one which gives best improvement in performance in the state being assessed [KAN92]
3) the one which gives best improvement in performability.

In 2), performance is optimised in a state of failure in order to obtain the best performance in that state itself. As a result, it is possible that performance in another state of failure is very bad. In contrast, in 3), in evaluation of performability, performance in all states is taken into consideration, that is the topology is chosen, which, on average, performs best in all the states in which the network can be. Each state is weighted by the probability that it will operate in such a configuration, and the residence time during which it is in that state, which are dictated by the failure rate and mean repair time associated with the components.

4.3 Definition of the Performability-based descent factor

If choice of topology is performability-based, then in the iterations of the *fault-tolerance augmentation* stage, the performability-based criterion, or descent factor [BAR92], is given by $\frac{\Delta\text{performability}}{\Delta\text{cost}}$.

$$\frac{\Delta\text{performability}}{\Delta\text{cost}} = \frac{\text{Perf}_{n+1}(b,t) - \text{Perf}_{n}(b,t)}{\text{Cost}_{n+1} - \text{Cost}_{n}} \qquad (4.1)$$

where $\text{Perf}_x(b,t)$: performability associated with the network after x iterations over an observation period $(0, t)$,

and Cost_x : cost of the network after iteration x.

$\text{Perf}_x(b,t)$ = P[average accumulated reward $ACR(t) \leq b$] over $(0, t)$.

Performability could also be evaluated over a more specific range of performance levels (a, b) where $\text{Perf}_x(a,b,t)$ is P[$a \leq$ average accumulated reward after x iterations $\leq b$] over an observation period $(0, t)$. The performability measures are determined using the *randomisation technique* described in Chapter 3.

In Figure 4.7, $\text{Perf}_n(b,t)$ and $\text{Perf}_{n+1}(b,t)$ correspond to typical profiles of the distribution of average accumulated reward associated with the network after n and $(n+1)$ iterations respectively. The difference in performability at reward equals b is shown in the

figure. If performability is evaluated over a range of rewards from a to b, it is computed as the difference between P[$ACR(t) \leq b$] and P[$ACR(t) \leq a$] for the distribution obtained at each iteration.

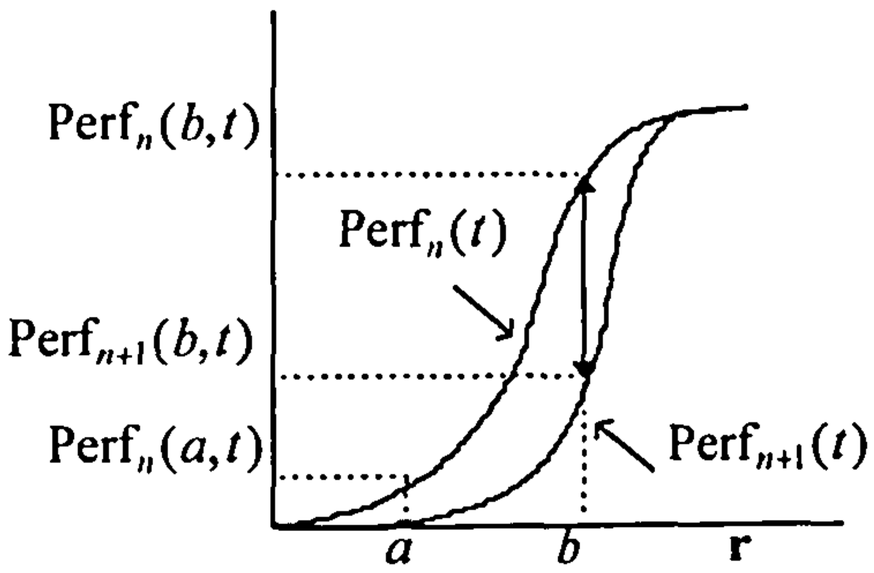


Figure 4.7 -Typical profiles for performability distributions after n and (n + 1) iterations

In a case in which the budget is assumed to be unlimited, cost need not be taken into account in the descent factor. In the more practical situation, the cost of the network has to be considered. Two definitions of cost are proposed in Sections 4.5.1 and 4.5.2, whereby one refers to cost of the network in terms of its components alone and the other includes additionally the cost of repair expected to be incurred over a certain observation period.

4.4 Reward definition

In order to evaluate performability, performance levels have to be mapped on to rewards. Ideally, there should be a reward for each possible performance level but the higher the number of rewards, the more computationally intensive determination of performability becomes. If there are K+1 rewards, at n transitions, the number of possible ways of assigning the rewards to the intervals is given by $\binom{K+n+1}{n+1}$. In the design examples considered in the thesis, only three rewards have been used. As explained in Chapter 3, the

implication of using fewer rewards than the number of attainable performance levels is that the distribution of average accumulated reward gives an indication of the distribution of average performance, but they do not map exactly on to each other. The rewards can be defined in two ways.

4.4.1 Reward definition 1

The first definition of reward is as follows: x is the performance variable associated with each state, and $\rho_3 < \rho_2 < \rho_1$, assuming that the lower the value of *level_x*, the better the performance level, hence the higher the reward. *Level_a* is less than *level_b*.

$$If\ x < level_a,\ reward = \rho_1$$

$$If\ level_a \leq x < level_b,\ reward = \rho_2$$

$$If\ level_b \leq x,\ reward = \rho_3.$$

4.4.2 Reward definition 2

In some cases, using the above reward definition, two or more topologies can be associated with the same reward structure. For example, any performance level above *level_b* is associated with reward ρ_3. If *level_b* is taken to be 500, then a performance level of 1000 and one of 5000 are both mapped on to the same reward. In order to differentiate between them, it is necessary to consider the range of performance levels that is being covered. Then, the second definition of reward is used. The maximum *(upper)* and minimum *(lower)* performance levels are first identified as follows.

$$If\ \frac{x - lower}{upper - lower} < level_c,\ reward = \rho_1$$

$$If\ level_c \leq \frac{x - lower}{upper - lower} < level_d,\ reward = \rho_2$$

$$If\ level_d \leq \frac{x - lower}{upper - lower},\ reward = \rho_3.$$

Note that if performance is identified with a measure such that the higher its value, the better the performance, then the reverse relationship holds in assignment of rewards. In *definition 1*, the higher the level of x, the higher the reward. In *definition 2*, the higher the value of $\frac{x - lower}{upper - lower}$, the higher the reward.

4.4.3 Application of the two definitions in reward assignment

The following example is used to demonstrate the use of the definitions of rewards. Let the system under consideration be a network which consists of five links, labelled 1 to 5, and let it be assumed that the links fail one at a time. Assume that application of two separate capacity augmentations need to be investigated, thereby leading to two topologies: network1 and network2. Let the performance levels of these two topologies in their different states be as given in Table 4.1, where performance is associated with a measure such as delay. In this case, the lower its value, the better the performance delivered by the network. Assume that three rewards denoted by $\rho_1 > \rho_2 > \rho_3$ are used, where the lower the performance, the higher the reward. The rewards associated with each state, derived using both definitions, are also given in Table 4.1.

Let rewards be defined according to *definition 1* as follows, with the performance level being denoted by x.

If $x < 0.01$, *reward* $= \rho_1$

If $0.01 \leq x < 0.02$, *reward* $= \rho_2$

If $0.02 \leq x$, *reward* $= \rho_3$.

Configuration	Performance of Network1	Reward Defn. 1	Reward Defn. 2	Performance of Network2	Reward Defn. 1	Reward Defn. 2
Fully operational	0.004	ρ_1	ρ_1	0.005	ρ_1	ρ_1
Link 1 fails	0.018	ρ_2	ρ_2	0.011	ρ_2	ρ_1
Link 2 fails	0.025	ρ_3	ρ_2	0.050	ρ_3	ρ_3
Link 3 fails	0.030	ρ_3	ρ_2	0.060	ρ_3	ρ_3
Link 4 fails	0.008	ρ_1	ρ_1	0.007	ρ_1	ρ_1
Link 5 fails	0.030	ρ_3	ρ_2	0.025	ρ_3	ρ_2

Table 4.1 - Reward definitions for Network1 and Network2

When rewards are specified according to *definition 1*, the two topologies are seen to be associated with the same reward structure, as shown in the *Defn. 1* column of Table 4.1, and appear to be equivalent. These rewards do not reflect the different degrees of degradation caused when for example, links 2 or 3 fail. Therefore, the range of performance levels being covered needs to be considered to differentiate between the two networks.

In *definition 2*, the upper and lower levels of performance are set to be 0.06 and 0.002 respectively. The values of *level_c* and *level_d* are chosen to be 0.2 and 0.5. Using these values, the rewards associated with the states in which the network can be, are worked out. The results are as shown in the *Defn. 2* column of Table 4.1.

The two networks are now associated with two different reward structures, which can be used in evaluation of their performability. These rewards give a better indication of the degree of degradation caused when a failure occurs. In generation of results in this thesis, although *definition 1* is generally used in the assignment of rewards, when comparison of networks with similar reward structures is required, *definition 2* comes into play in order to differentiate between them.

4.5 Definition of the cost function

Two definitions of the cost function are considered. In both cases, the cost of the network is defined in terms of the cost of its component nodes and links. In the second definition, however, the cost of repair expected to be incurred over a given observation period is also taken into consideration.

4.5.1 Cost function: Definition 1

The first definition of cost is based on the definition used in [KAN92]. A link is assumed to consist of a number of channels of fixed capacity so that the capacity of a link is determined by the number of channels constituting it. Cost of a link is taken to be linearly proportional to its capacity, as well as its physical length. A node consists of switching components and buffers. As used by Kang and Tan [KAN92], the nodal hardware switching cost is assumed to be proportional to the square of the number of channels connected to it, in order to reflect the increase in complexity of switching components as more channels are added. The cost of buffers at a node is taken to be linearly proportional to the size of the buffers, which is a function of the number of channels connected to the node, the capacity of those channels, and the rates at which packets arrive and leave those buffers. Allocation of buffer capacity has to take into consideration the statistics of the arrival stream of packets, the rate at which these packets can be serviced, and directed on to outgoing lines. For the sake of simplicity, it is assumed in this thesis that buffers have a certain maximum capacity and a buffer of that capacity, is reserved for any outgoing or incoming channel at a node.
The cost function is thus defined as

$$\begin{aligned} D_1(N) &= \sum_{j=1}^{L} D_1(N_j) \\ &= \sum_{j=1}^{L} \left[\delta_j + \alpha_j N_j + \gamma_j N_j + \beta_j N_j^2 \right] \end{aligned} \tag{4.2}$$

where

N_j is the number of channels constituting simplex link j,

δ_j is the initial set-up cost of link j,

$\alpha_j = s_j L_j$ where s_j is the cost per channel per km for a simplex link j,

and L_j is length of the link,

γ_j is the cost per buffer per channel for link j and

β_j represents the switching costs per link j.

4.5.2 Cost function : Definition 2

Given that performability is evaluated over a period of time, it is meaningful to consider the cost of maintaining the network, besides the cost of setting it up. In Chapter 3, the algorithm by Iyer *et al.* [IYE86] was described as an algorithm for evaluating the moments of accumulated reward, in a context where reward represents performance. In this thesis, it is proposed to use this algorithm to compute the average cost of repair expected to be incurred over a period of time. If the reward associated with a state of failure represents the average cost of repair per unit time of the component which has failed, the mean accumulated reward over a given observation period corresponds to the average cost of repair of the overall network during that time. Furthermore, by using this algorithm, the cost of repair also takes into account the time taken to do so, which is given by the length of time for which the network resides in a state of failure. Cost is then defined as

$$D_2(N) = \sum_{j=1}^{L} D_1(N_j) + \text{Average repair cost over time duration } (0,t) \qquad (4.3)$$

The algorithm used is affected neither by the number of distinct rewards used, nor by the range of these rewards so that it is not necessary to map the repair cost for each component on to a reward. If a mapping is done, the average cost of repair is computed as the mean accumulated reward over time period $(0,t)$, which can be mapped back to an average repair cost for the whole system, given that the mapping is one-to-one.

4.6 Choice of performance measure and routing algorithm

Choosing the routing algorithm to apply to the network being designed, and the performance measure to assess the network, are very important. In order that performability shall reflect

the fault-tolerance of the system, it is necessary that the performance measure is such that it shows degradation in states of failure. For instance, under conditions of non-optimal routing prior to re-routing, it is possible that the network can have a better performance after re-routing following failure of a component. Also, when node failures occur, the resulting network can have a better average delay if packet loss is ignored. The examples, which are given later, will show how these cases can occur.

4.6.1 Network simulation

The examples to be discussed are based on discrete event simulation of simple networks, whereby the system is modelled in terms of events which cause the state of the network to change. Such events are for instance the arrival of a packet at a node, or the transmission of a packet over a link. A network is characterised by a number of different parameters which can be classified in the following categories:

1. the topology of the network - the number of nodes and links and their configuration,
2. routing algorithms for packets in the network and
3. other parameters such as transmission rates, arrival rates of packets and average packet length.

4.6.2 Simulation model

A packet is treated as an entity which is created when it enters the network and is discarded once it has reached its destination. Nodes and links are considered to be servers. Each node is modelled as consisting of a processor and an output buffer for each outgoing link from the node. As soon as a packet reaches a node, it is assumed to be processed in negligible time and is added to the FIFO output buffer at the node for the link over which the packet needs to be transmitted. The link is modelled as a server which is used to transmit packets as long as they are available in its corresponding buffer. Links are considered to be unidirectional.

Essentially, the input data characterises the configuration of the network. The simulation is run for a fully operational configuration to obtain a performance measure. Then, the input data is changed to indicate failure of component x, and the corresponding

performance measure is noted. Only single failures are allowed in order to avoid problems resulting from consideration of a large state space.

The following assumptions have been made. Bi-directional communication between a pair of nodes is ensured by two unidirectional links, carrying traffic in opposite directions. Failure of a link is assumed to be treated as failure of both links between the connecting pair of nodes. Any packets requesting that link are routed to their destination on a different set of links if possible. If no such set exists, the packets are discarded. On a node failure, all links connected to that node are assumed to have failed; and any packets destined for that node are discarded. This is to avoid infinite delays.

4.6.3 Example 1: Coping with node failures

In this section, the issue of choosing a performance measure suitable for evaluating networks where node failures occur is discussed. Let the performance measure used be the average end-to-end delay of packets. Based on the above assumptions, it is possible that, given a network of n nodes and m links, the failure of a node will yield a configuration of (n - 1) nodes and k links, which may have a better average network delay than the fully operational one. The network thus appears to be performing better if the number of packets lost is ignored. Therefore, there is a need to take into account both the average network delay and the average rate of discarding packets to show degradation caused by breakdowns in the network. The example below is used to show this.

4.6.3.1 Network topology

Consider the network shown in Figure 4.8, which consists of 6 source / destination nodes and 16 unidirectional links connecting the nodes in the arrangement illustrated.

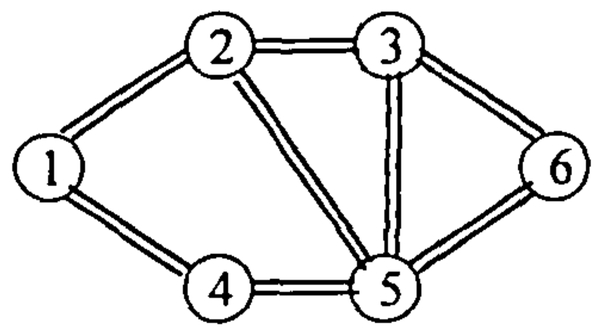


Figure 4.8 - Configuration of a 6-node network

The above network is simulated with the following input parameters:

1. Packets arrive at each node at a rate of 50 packets/s according to a Poisson process.
2. Each link has a transmission capacity of 64000 bits/s.
3. Packet lengths are exponentially distributed with a mean of 256 bits.

Varying the input parameters would give different results in terms of average end-to-end delays, or number of packets lost. For instance, an increase in the capacity of the links, with other parameters remaining constant, entails a decrease in the average packet delays. However, given that the point of this example is to investigate an appropriate way of measuring the performance of the network where node failures can occur, results are not presented for different sets of input parameters.

4.6.3.2 Routing using Dijkstra's algorithm

When a packet arrives at a node, it is routed to its destination via the shortest path available. The shortest path is determined using Dijkstra's algorithm. Each link is associated with a label, or length, which in the simulation is taken to be the average packet delay across it. The average packet delay includes the average waiting time in the output buffer at the node before transmission and the transmission time over the link. Each node is then associated with a label which corresponds to the sum of the length of the incoming link and the label of the node at the other end of the link.

The algorithm can be formalised as follows. Let each node in the network be referred to as a vertex x_i and let the label attached to it be given by $l(x_i)$. s is the node from which the shortest path to a specific node, or to all the other nodes in the network, has to be determined.

Initialisation

Step 1. Set $l(s) = 0$ and mark the label as permanent. Set $l(x_i) = \infty$ for all $x_i \neq s$ and mark these labels as temporary. Set $p = s$.

Updating of labels

Step 2. For all x_i adjacent to p and which have temporary labels, update the labels according to

$$l(x_i) = \min [\, l(x_i),\ l(p) + c(p,\ x_i) \,]$$

where $c(p,\ x_i)$ is the label associated with link between nodes p and x_i.

Fixing a label as permanent

Step 3. Of all temporarily labelled vertices find x_i^* for which $l(x_i^*) = \min[l(x_i)]$.

Step 4. Mark the label of x_i^* as permanent and set $p = x_i^*$.

Step 5.

(i) If only the path from s to t is required,

If $p = t$, $l(p)$ is the required shortest path length. Stop.

If $p \neq t$, go to Step 2.

(ii) If the path from s to every other vertex is required,

If all the vertices are permanently labelled, then the labels are the lengths of the shortest paths. Stop.

If some labels are temporary, go to Step 2.

4.6.3.3 Application of Dijkstra's algorithm in the simulations

A routing table is maintained at each node where, given the destination node of a packet, the next node to which it should be forwarded and the corresponding link required can be looked up. This table is initialised at the beginning of the simulation. The shortest path algorithm is applied to a network where all link lengths are equal to 0 and results of that first run, although quite arbitrary, are used for routing packets for the first part of the simulation. Then, at regular intervals, the routing table is updated by determining the shortest paths based on current link lengths, or average packet delays, over each link. The intervals have been kept long enough to avoid any major problems due to oscillation between paths.

4.6.3.4 Performance measure from simulation

The results gathered from the simulation runs have to be such that they can be used to provide a comparison between the different configurations. In a fully operational network and in configurations where link failures occur, there is no loss of packets. If a node fails, packets are inevitably lost. Assuming that average network delay is the performance measure determined from a simulation run, then as indicated earlier, it is possible that the configuration resulting from a node failure will give a lower average network delay than the one obtained for the fully operational network. This does not mean that this configuration performs better because loss of packets is a clear sign of degradation.

Application of Dijkstra's algorithm to the network in Figure 4.8 gives the following results shown in Table 4.2 when node and link failures occur. The link labels (*i-j*) represent the links connecting nodes *i* and *j*. The average rate of discarding packets, in the presence of node failures, corresponds to the average rate of losing packets which are generated at the node which has failed, and discarding those which are destined for that node.

Configuration	Average network delay (s/packet)	Average discard rate of packets (packets/s)
Fully operational	0.0072	0
Node 1 fails	0.0063	99.86
Node 2 fails	0.0082	100.43
Node 3 fails	0.0075	99.86
Node 4 fails	0.0069	100.26
Node 5 fails	0.0101	100.05
Node 6 fails	0.0063	100.29
Links (1-2) fail	0.0087	0
Links (1-4) fail	0.0080	0
Links (2-3) fail	0.0081	0
Links (2-5) fail	0.0076	0
Links (3-5) fail	0.0081	0
Links (3-6) fail	0.0077	0
Links (4-5) fail	0.0096	0
Links (5-6) fail	0.0080	0

Table 4.2 - Initial performance of the 6-node network

From Table 4.2, when nodes 1, 4, or 6 fail, average network delay is reduced as compared with the value obtained when all the nodes are operational so that the network appears to be performing better. However, the number of packets discarded shows that degradation has occurred. When comparing the configurations where link failures occur with those where node failures occur, the degradation caused is not reflected by the average network delay. No packets are discarded in the presence of link failures. The need to consider both the average network delay and average rate of discarding packets leads to the following approach being adopted in the simulation and consideration of the results.

If a number of networks are to be compared with respect to two variables, it is easier to do so if one of the variables is kept constant for all of them. In this case, the average network delay is chosen as the variable which should remain constant. A maximum

allowable delay is imposed on every network; and any packets exceeding this value whilst in transit through the network are discarded. The maximum delay is varied for each configuration so that the average network delays obtained for the network in its different configurations are all the same. Then the average discard rate of packets is used as performance variable to compare the various networks. It should be noted that this results in packets being discarded, irrespective of whether failure has occurred or not.

In brief, the situation is as follows. The input traffic to the network is kept constant; and the average network delay does not vary. The only result which changes is the average discard rate which is then taken as the performance measure.

4.6.3.5 Simulation results

Based on the above approach, the simulation of the network in Figure 4.8 was carried out with the network in every possible configuration as dictated by failure of only one component at a time. Results are as shown in Table 4.3.

Performance is measured in terms of the average rate of discarding packets, which now includes packets being discarded when they exceed the maximum transit time imposed as well as those usually discarded in the presence of node failures. The average network delay was fixed at 0.0063 s. For each configuration, the maximum allowable delay is varied so that the average network delay is kept at a level of 0.0063 s. The variations and the resulting effects on the average discard rate of packets are as shown in Table 4.3.

Configuration	Maximum allowable delay (s/packet)	Average discard rate of packets (average delay is 0.0063s) (packets/s)
Fully operational	0.0210	4.46
Node 1 fails	10.0	99.86
Node 2 fails	0.0180	107.85
Node 3 fails	0.0195	104.14
Node 4 fails	0.0245	101.03
Node 5 fails	0.0155	121.78
Node 6 fails	10.0	100.29
Links (1-2) fail	0.0170	16.75
Links (1-4) fail	0.0180	10.21
Links (2-3) fail	0.0180	10.73
Links (2-5) fail	0.0185	7.62
Links (3-5) fail	0.0180	11.10
Links (3-6) fail	0.0185	7.54
Links (4-5) fail	0.0165	25.98
Links (5-6) fail	0.0180	10.42

Table 4.3 - Performance of the 6-node network
when a maximum delay is imposed.

Under conditions of fixed average network packet delay, the average rate of discarding packets is used as means of assessing the performance of the network. Even though packets are lost in the fully operational state, such a measure shows the degradation which occurs in the presence of node failures, which is not apparent in values obtained for the average network delays when no maximum delay is imposed. It is also a better indication of degradation caused when link failures occur, as compared with that resulting from node failures. When comparing the various configurations where links fail, the degree of degradation observed in terms of average network delay is reflected in the degree of degradation obtained in the average rate of discarding packets. For example, failure of links

between nodes 4 and 5 causes most degradation in average delay, which is reflected by the highest rate of discarding packets.

4.6.3.6 Conclusion from example 1

The results obtained using Dijkstra's algorithm and the above means of assessing performance are positive in the sense that, as expected, there is degradation as a result of failure of a component in the network. Given that the capacity of a network and the total input traffic are kept constant, it is expected that the network performance worsens under conditions of failure. Therefore, in cases where node failures have to be taken into consideration, performance can be evaluated in the way already described. The randomisation technique can then be applied, where the performance variable, *average rate of discarding packets*, is mapped on to rewards in order to generate performability measures.

4.6.4 Example 2: Case of use of non-optimal routing - probabilistic routing

In this example, the combined effect of applying a non-optimal routing algorithm to a symmetrical network is considered. Probabilistic routing, which does not attempt to optimise any performance measure, is used.

4.6.4.1 Network topology

The network under analysis is shown in Figure 4.9, and consists of 6 nodes and 24 links, where three of the nodes are source / destination nodes (nodes 1, 2, and 3), and the remaining three (nodes 4, 5, and 6) are transit nodes.

1. Source / destination nodes are nodes via which packets enter and leave the network.
2. Transit nodes are only used as intermediate nodes in routing.

The links are classified into three categories: a *primary* link connects two source / destination nodes. A *secondary* link corresponds to one between a source / destination node and a transit node, and a *tertiary* link connects two transit nodes.

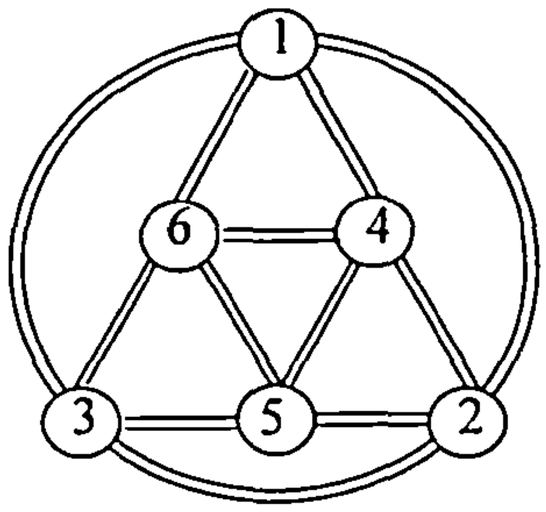


Figure 4.9 - Configuration of a symmetrical 6-node network

The input parameters used in simulation of the above network and in the generation of results are as follows:

1. Packets arrive at each source / destination node at a rate of 200 packets/s according to a Poisson process.
2. Each link has a transmission capacity of 64000 bits/s.
3. Packet lengths are exponentially distributed with a mean of 256 bits.

4.6.4.2 Essence of random routing

From a given node, each outgoing link has associated with it a certain probability, which is the probability that any packet arriving at that node will be transmitted to its destination via that link. When a packet arrives at a node, a random number is drawn from the uniform distribution on the range (0,1) and, depending on the value, a link is chosen for the packet to be routed to the next node. In so doing, the packet can reach its destination after one or more hops, and can even keep looping endlessly round the network without ever reaching its destination. The issue of routing in such a way as to optimise performance does not arise.

4.6.4.3 Simulation results

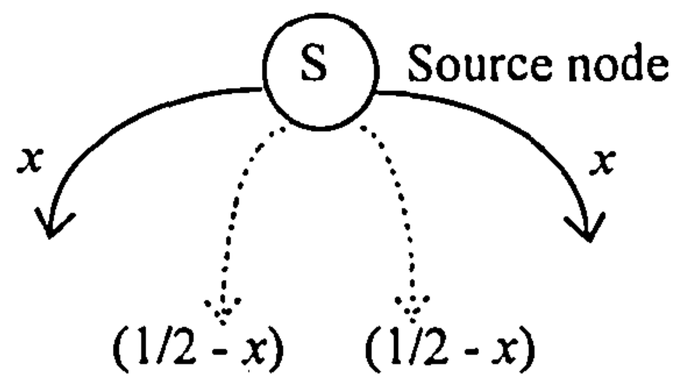
The network in Figure 4.9 has been analysed in the fully operational configuration as well as under conditions of failure, with respect to different probabilities of routing traffic via a

primary and a secondary link at each source node. Performance is measured as the average transit time of packets in the network. Only single link failures are investigated. Variations in these probabilities have a direct bearing on the number of hops after which a packet reaches its destination, and hence the performance of the network.

Consider the network in its fully operational configuration. At each source node, an external stream of packets arrives, which is to be routed to the two other destination nodes. Let all packets be transmitted via the secondary links, that is, via a transit node, with the average transit time being t_1. For example, packets arriving at source node 1 destined for node 2 may be routed via transit nodes 4 or 6. If the fraction of packets transmitted across the primary links increases, the average transit time is expected to decrease as a result of the higher proportion of packets reaching their destination after one hop. In the example considered, more packets are transmitted directly to node 2 via a primary link. However, as that fraction increases further, it also means that the fraction of packets being routed to their destination in two hops via one source / destination node increases. Then the average transit time starts to increase. This corresponds to packets which are routed to node 2 via node 3.

Therefore, there is an optimal way of splitting the packets between the possible outgoing links from each node which gives a minimum transit time. Note that if maximum allowable delay across the network is fixed at α, the discard rate of packets is expected to follow the trend shown by the average transit time of a packet in the network.

The same argument applies to networks in their various failure configurations. The changes of average transit time with variations in the probability of routing traffic via a primary link are shown in Figures 4.11(a), 4.11(b), 4.12, and 4.13. It is assumed that the same probabilities are applied at each source node. Furthermore, the probability axis corresponds to the value x in the figure below, which shows a source node, and its outgoing links with the probabilities associated with them.



——— : primary link

..... : secondary link

Figure 4.10 - Distribution of traffic at a source node

Figures 4.11(a) and 4.11(b) show the variation of average transit time as the probability of routing packets via a primary link increases for a fully operational network and a configuration where a single primary link has failed. Note that failure of a link is treated as failure of bi-directional communication between the connecting nodes.

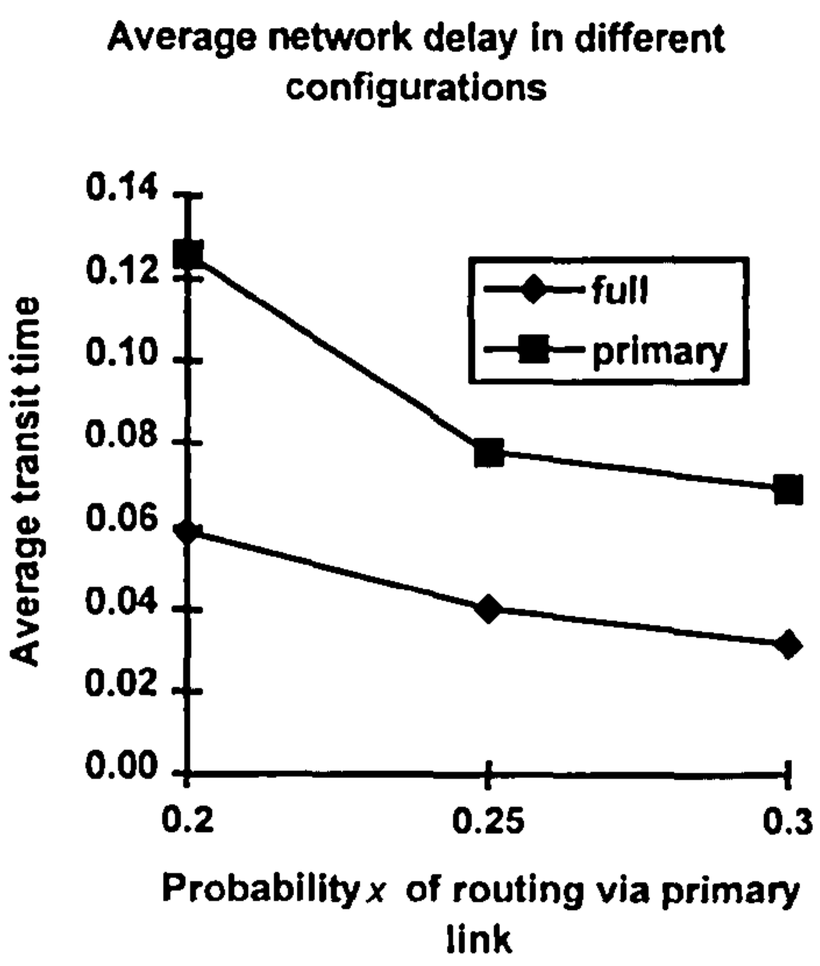


Figure 4.11(a) - Comparison between fully operational state and configuration where primary link has failed over range x between [0.2, 0.3]

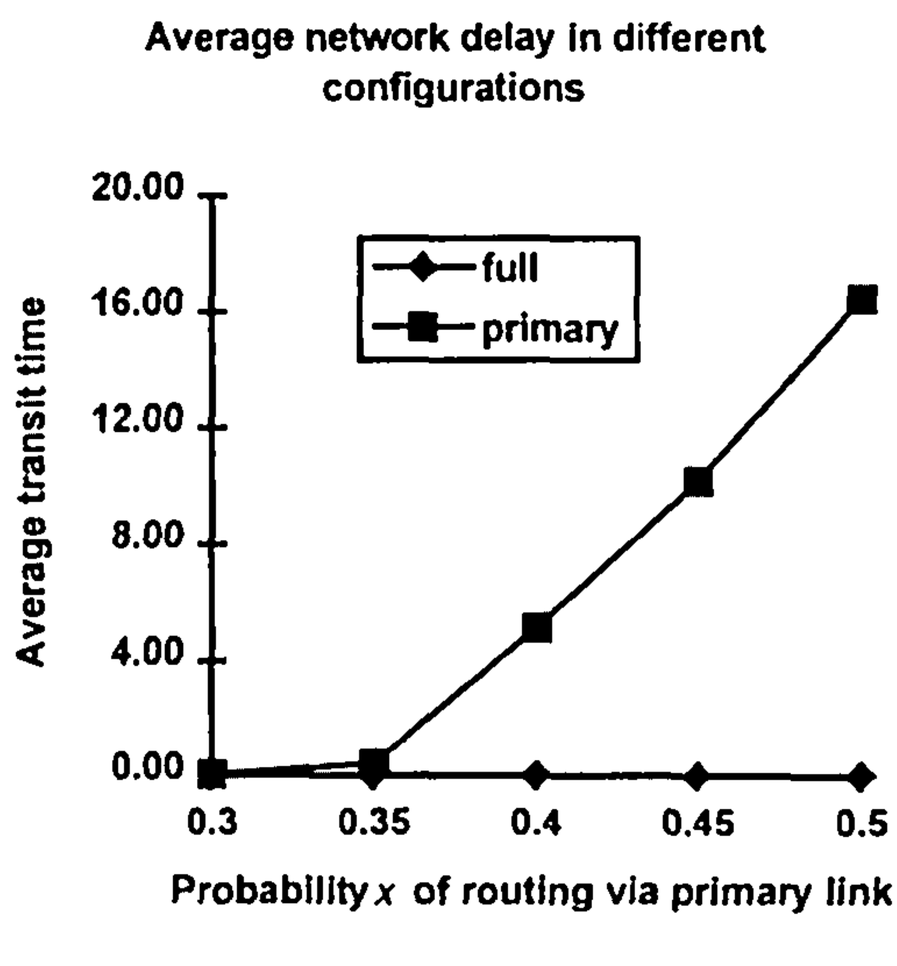
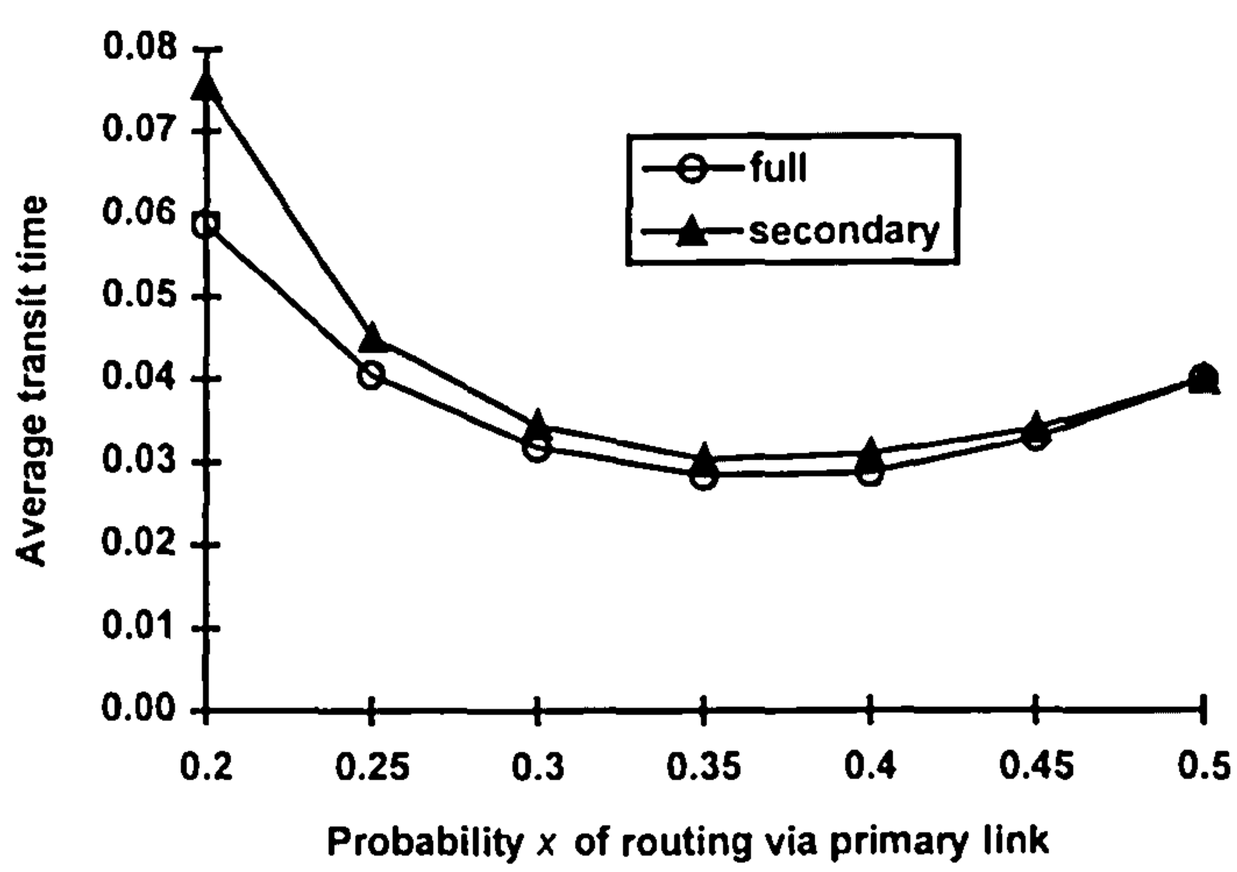


Figure 4.11(b) - Comparison between fully operational state and configuration where primary link has failed over range x between [0.3, 0.5]

full : *network in its fully operational state.*

primary : *network where links connecting nodes* 1 *and* 2 *fail.*

Figure 4.12 shows similar comparisons between a fully operational network and one where links connecting a source / destination node and a transit node fail.

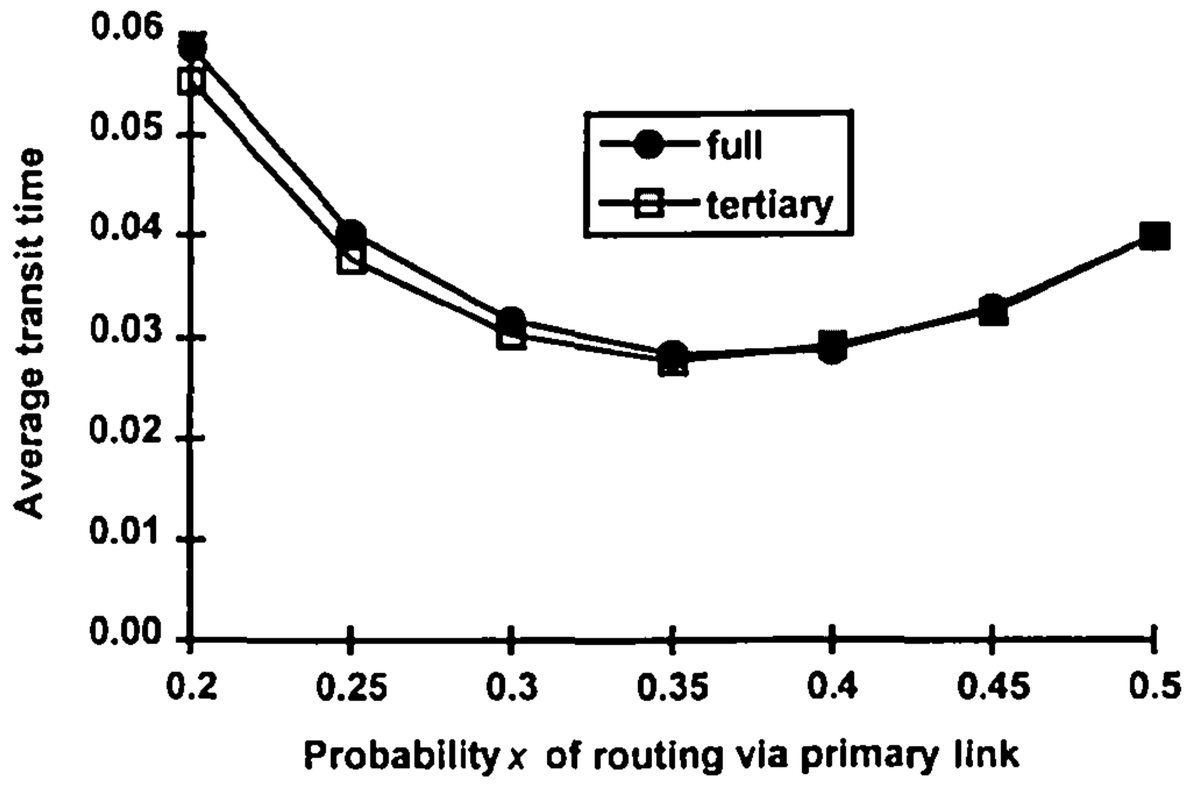


full : *network in fully operational conditions*

secondary : *network where link between nodes* 1 *and* 4 *has failed.*

Figure 4.12 - Comparison between network in a fully operational state and in a configuration where a secondary link has failed

Figure 4.13 shows results when comparison is done between the network in a fully operational state and in a configuration where a tertiary link has failed.



full : *network in fully operational conditions*
tertiary : *network where links between nodes 4 and 6 fail.*

Figure 4.13 - Comparison between network in a fully operational state and in a configuration where a tertiary link has failed

For each possible configuration, there is an optimal way of applying random routing to get the best performance. In other words, there is a best way of assigning the probabilities to the outgoing links from each node. However, it is expected that the best non-failed network would outperform the best network in a state of failure. Such results are obtained when comparing the fully operational network with configurations where primary or secondary links fail, as shown in Figures 4.11(a), (b), and 4.12.

When tertiary links fail, it appears, from Figure 4.13, that the network performs better than it does when all components are functioning. The reason this happens is because tertiary links introduce additional hops in the path of a packet from its source to its destination. They also contribute to a higher degree of endless looping of packets in the

network. Therefore, upon removal of such links, these features are reduced, thereby leading to an improvement in the performance of the network. At the point where all packets are routed via primary links only, a fully operational network and one where there is failure between two transit nodes are equivalent with the same performance level being obtained.

4.6.4.4 Conclusion from example 2

This example shows the effect of using a non-optimal routing algorithm, which does not attempt to optimise the performance of the network. Even though the network in a fully operational configuration is equipped with more capacity, the additional resources are not being used in such a way as to improve the performance of the network. The fact that the routing is being done without any regard to performance also contributes to this problem. In view of these results, it is sensible to use an optimal routing mechanism, which attempts to route packets in the network so as to optimise performance. It makes no sense to try and improve the fault-tolerance of a network where a sub-optimal routing algorithm is being applied through more connectivity and capacity augmentation if the same effect can be achieved by changing the routing at the very outset.

4.6.5 Optimal routing - GPM based [BER87]

The essence of optimal routing is to direct traffic exclusively along paths which are shortest with respect to some link lengths that depend on the flows carried on the links. In the routing algorithm used in the proposed design method, a Gradient Projection based Method (GPM) is applied to achieve this by minimising a particular cost function which is in the form $\sum_{(i,j)} D_{ij}(F_{ij})$

where $D_{ij}(F_{ij}) = \dfrac{F_{ij}}{C_{ij} - F_{ij}}$ (4.4)

C_{ij} is the capacity of link (i, j) and

F_{ij} is the total flow carried by link (i, j) given by

$$F_{ij} = \sum_{\substack{\text{all paths } p \\ \text{containing } (i,j)}} x_p \qquad \text{where } x_p \text{ is the flow of path } p.$$

For every origin-destination pair w, there are the constraints

$$\sum_{p \in P_w} x_p = r_w$$

$$x_p \geq 0, \text{ for all } p \in P_w.$$

where r_w is the given traffic input of the origin-destination pair w and

P_w is the set of directed paths of w.

The problem expressed in terms of the unknown path flow vector $x = \{ x_p \mid p \in P_w, w \in W\}$ is written as

$$\text{minimise } \sum_{(i,j)} D_{ij}\left[\sum_{\substack{\text{all paths p} \\ \text{containing (i,j)}}} x_p\right], \tag{4.5}$$

$$\text{subject to } \sum_{p \in P_w} x_p = r_w, \text{ for all } w \in W$$

$$x_p \geq 0, \text{ for all } p \in P_w, \ w \in W$$

Optimal routing is characterised in terms of derivatives of the functions D_{ij}. Each D_{ij} is assumed to be a twice differentiable function of F_{ij} and is defined in an interval [0, C_{ij}), where C_{ij} is a positive number typically representing the capacity of the link. The first and second derivatives of D_{ij} are denoted by D'_{ij} and D''_{ij}, respectively and are assumed strictly positive for all F_{ij} in [0, C_{ij}).

Denote the cost function by $D(x)$ where

$$D(x) = \sum_{(i,j)} D_{ij}\left[\sum_{\substack{\text{all paths p} \\ \text{containing } (i,j)}} x_p\right] \tag{4.6}$$

and $\partial D(x)/\partial x_p$ as the partial derivative of D with respect to x_p. Then

$$\frac{\partial D(x)}{\partial x_p} = \sum_{\substack{\text{all links (i,j)} \\ \text{on path p}}} D_{ij}^{'}, \tag{4.7}$$

where the first derivatives $D_{ij}^{'}$ are evaluated at the total flows corresponding to x. The derivative $\partial D(x)/\partial x_p$ is the length of path p when the length of each link (i, j) is taken to be the first derivative $D_{ij}^{'}$ evaluated at x. The derivative $\partial D(x)/\partial x_p$ is called the *first derivative length of path p*.

Optimal routing results only if flows travel along minimum first derivative length (MFDL) paths for each origin-destination pair. Equivalently, a set of path flows is strictly sub-optimal only if there is a positive amount of flow that travels along a non-MFDL path. This suggests that sub-optimal routing can be improved by shifting flow to an MFDL path from other paths for each origin-destination pair. A number of methods exist to determine how to shift the flow from some paths to the shortest path. Generally, they solve computationally the optimal routing problem by decreasing the cost function through some incremental changes in path flows.

Given a feasible path flow vector $x = \{x_p\}$, x can be changed along a direction $\Delta x = \{\Delta x_p\}$ where the two main requirements imposed on the direction Δx are

1. Δx should be a feasible direction in the sense that small changes along Δx maintain the feasibility of the path flow vector x and
2. Δx should be a descent direction in the sense that the cost function can be decreased by making small movements along the direction Δx starting from x.

The *Frank-Wolfe* (*Flow Deviation*) *Method* is one way of implementing the idea of having incremental changes along feasible descent directions. Another class of feasible direction algorithms for solving the optimal routing problem consists of the projection methods. These methods tend to be faster than the *Frank-Wolfe Method*. They are based on shortest paths and determine a minimum first-derivative length path for each origin-destination pair at each iteration. An increment of flow change is calculated for each path on

the basis of the relative magnitudes of the path lengths and, sometimes the second derivative of the cost function. If the increment is so large that the path flow becomes negative, then the path flow is simply set to zero.

The routing algorithm used throughout the design consists of the application of a Gradient Projection based Method to minimise the cost function referred to in equation (4.4) and achieve optimal distribution of flows of traffic. If each link is modelled as an *M*/*M*/1 system, the cost function corresponds to the average number of packets in the network. Average network delay per packet is then expressed as $\frac{1}{\gamma}\sum_{all\ links}\frac{F_{ij}}{C_{ij}-F_{ij}}$, where γ is the total arrival rate into the network.

This GPM based routing algorithm, where the shortest path problem is solved using the Pape's modification of Bellman's algorithm [BER84], is applied to the networks in Figures 4.8 and 4.9 to show how performance, in terms of average network delay per packet changes. Only link failures are considered. Results are as shown in Tables 4.4 and 4.5.

Configuration	**Average delay using Dijkstra's algorithm (s/packet)**	**Average delay using GPM-based routing algorithm (s/packet)**
Fully operational	0.0072	0.0070
Links (1-2) fail	0.0087	0.0087
Links (1-4) fail	0.0080	0.0079
Links (2-3) fail	0.0081	0.0079
Links (2-5) fail	0.0076	0.0075
Links (3-5) fail	0.0081	0.0078
Links (3-6) fail	0.0077	0.0075
Links (4-5) fail	0.0096	0.0095
Links (5-6) fail	0.0080	0.0079

Table 4.4 - Comparison between the GPM based routing algorithm and Dijkstra's algorithm for the network in Figure 4.8

In Table 4.4, the link labels (i - j) correspond to links connecting nodes i and j. From the table, it appears that the GPM-based routing algorithm gives better results. This is due to the nature of the implementations. As the time between consecutive updates of the routing table when Dijkstra's algorithm is used tends to zero, the results obtained should converge towards the results obtained when the GPM is used [GAF87].

Configuration	**Best average delay using probabilistic routing (s/packet)**	**Average delay using GPM-based routing algorithm (s/packet)**
Fully operational	0.0284	0.0064
Primary links fail	0.0692	0.0083
Secondary links fail	0.0303	0.0065
Tertiary links fail	0.0278	0.0064

Table 4.5 - Comparison between the GPM based routing algorithm and probabilistic algorithm for the network in Figure 4.9

In Table 4.5, due to symmetry of the network in Figure 4.9, results are not shown for all configurations arising from failure of each pair of links between two nodes. The failure configurations can be classified into three categories in which primary, secondary, or tertiary links fail respectively. Although single link failures are being considered, failure of any unidirectional link is treated as failure of both links between the connecting pair of nodes. The results given for the network when probabilistic routing is used correspond to the best values obtained when the probabilities of routing packets via primary links are varied. As shown, a definite improvement in average network delays is achieved on application of the GPM-based routing algorithm. Furthermore, in the fully operational network, packets are not actually routed via the tertiary links so that when they fail, the performance of the network is not affected. They are not used when secondary links fail, but some tertiary links are used for transmission in the presence of primary link failures.

4.7 Conclusions

In this chapter, a new design strategy, which aims at producing fault-tolerant networks, has been proposed. The objective network is one which can maintain its performance and is, as

far as possible, least affected by component failures. Design of such a network calls for a means of evaluating the network with respect to the performance it can deliver over a period of time, when components may fail and take time to be repaired. In Chapter 3, it was established that performability, which includes aspects of performance and reliability, gives an indication of the fault-tolerance of the system being evaluated. This leads to the proposed design technique being performability-based.

Furthermore, as explained in Chapter 2, design is a process made hard by the number of parameters and constraints that have to be taken into consideration. Gerla and Kleinrock addressed the overall design problem as three separate sub-problems, each of which can be solved on its own, in order to reduce its complexity. These are topological design, capacity assignment, and routing. In accordance with these, the design proposal has attempted to tackle the issues of topological design and capacity assignment while assuming that optimal routing is being used.

With reference to the schematic diagram in Figure 4.1, the main stages constituting the proposed design technique have been developed. The steps involved in execution of the three stages, the *initial topology generation* stage, the *increase in connectivity* stage, and the *fault-tolerance augmentation* stage have been discussed, with more emphasis being placed on the last stage. The first two stages are required to design an initial network, which meets basic reliability requirements. Then, the *fault-tolerance augmentation* stage can be brought into play to augment the network with a view of equipping the network adequately for it to deliver a good performance in its fully operational state, and an acceptable level in states of failure.

A number of decisions need to be taken in the process of iteratively augmenting the network. The state where optimisation is to be investigated needs to be selected. The suitable means of augmenting the network have to be identified, and finally the best topology has to be chosen. The issues that differentiate this approach from conventional design lie in the criteria used in choice of the state and the topology. Traditionally, optimisation is applied to the network, assuming all components are functioning, in order to achieve the best performance under such conditions. In the proposed strategy, firstly, the

state where augmentation is to be applied is chosen to be a failure state. The criteria used for selecting the state can be based on performance, or performability where each link is associated with a performability index. The performability index was introduced in Chapter 3 as a concept which is expected to give an indication of the relative importance of components in a system. The state of worst performance, or highest performability index, is therefore chosen.

In each state which corresponds to the network operating in a particular configuration, the distribution of traffic among the links produces a certain scenario. Application of the augmentation strategy needs to take into consideration the load on the links. The two augmentation mechanisms used are the addition of links and capacity augmentation. A set of rules has been proposed, as a means of assisting the designer in deciding where to add new links, or apply capacity augmentation. Different possibilities for augmenting the network result in a number of topologies, out of which the best one has to be chosen. In contrast with traditional design, the topology is chosen which, on average, performs well in all the states considered, where each state is weighted by the failure rate and mean repair time associated with it. This might not mean that the network performance is the best that can be had in the absence of failures; but taking into consideration performance in all the states, over a period of time, it delivers the best average performance. A performability-based descent factor is used to choose the topology, where evaluation of the network is done with respect to aspects of both performance and reliability. By using performability in this choice, the most fault-tolerant network is selected at the end of each iteration. Performability figures are evaluated using the *randomisation technique* [DES89].

The descent factor also needs to include the cost of the network. Two definitions of the way in which cost can be calculated have been proposed. In the first definition, cost is defined in terms of the links and nodes constituting the network. The second definition includes this fixed cost as well as the variable cost of maintaining the network, in terms of repairing components if they fail. The average repair cost of a network is determined using the algorithm by Iyer *et al.* [IYE86] for determining the moments of accumulated reward. Although the algorithm was developed to evaluate moments of performability, it has here

been applied to calculate the average repair cost, after considering the rewards assigned to each state to be repair costs per unit time.

In order to evaluate performability, performance levels have to be mapped on to rewards. Two definitions have been proposed for this mapping. In the first definition, performance levels are not assumed to be bounded by a specific range. Using such a definition, at times, fails to differentiate between networks which are associated with the same reward structure even though there is a considerable difference between the performance levels in the failure states. In this case, the second reward definition is required, where the range of performance levels covered is taken into consideration, thereby enabling a more accurate comparison of networks.

Prior to mapping performance on to rewards, the performance levels have to be determined for the given network. Choice of a suitable means of measuring the performance of the network, coupled with choice of an appropriate routing algorithm, may give rise to a number of problems. Two examples have been considered. In the first example, the question of finding a means of evaluating a network, which can be affected by node failures, is addressed. Results have shown that a combined measure of average network delay, and average discard rate of packets is required in order to reflect the degradation caused in the presence of node, and link failures. In the second example, application of a non-optimal routing algorithm leads to a situation where the network performed better when certain components failed. This feature was identified as being due to the non-optimality of the routing technique used, which did not attempt to optimise any performance measure. Consideration of this example led to a GPM-based routing algorithm being adopted in the proposed design technique. Traffic is expected to be optimally distributed in a way as to minimise the cost function, which corresponds to the average number of data units in the network. This is taken to be the means for measuring the performance of the network.

The design proposal has been developed; and its various constituent stages and aspects have been discussed. Results of application of this technique need to be considered in terms of the performance, performability, and cost of networks designed. The computational requirements of the proposal also have to be determined.

Chapter 5

Results of application of the performability-based design technique

5.1 Introduction

Now that the performability-based design method has been developed, its capability, relative merits, and shortcomings need to be established. In this chapter, the different aspects of the proposed design technique will be investigated.

Firstly, in Chapter 4, two definitions of the cost of the network were proposed, leading to consideration of two different performability-based descent factors. Cost of the network is evaluated in terms of the cost of its constituent components in the first definition. The second definition includes this value as well as the average cost of repair, expected to be incurred over a period of time. The two descent factors will be compared to establish their relative benefits with respect to applicability to a practical environment, where the amount of network investment may be a prime issue.

The rationale behind using a performability-based approach to design is to generate fault-tolerant networks. The concepts and arguments which differentiate this method from conventional design techniques are not network, or traffic dependent. The difference lies in the approach at augmenting the network. In this chapter, the design of a 10-node network is used to illustrate the capability of the proposed strategy. Owing to the nature of its underlying concepts, it is expected that the results of a comparison between the networks designed according to this approach and traditional design methods can be generalised to larger, and different networks. This is discussed further in Section 5.9.

The execution of the stages of the technique will be described as design of the 10-node network proceeds. The first two stages are discussed with reference to design of networks using physical distance, or traffic demands, between the nodes as a criterion for

choosing how to connect the nodes. Most emphasis is placed on the *fault-tolerance augmentation* stage, where different aspects need to be investigated. In this iterative stage, two decisions have to be taken. A state has to be chosen to investigate possibilities of augmentation at the beginning of an iteration, and the best topology is selected at the end of the iteration. Using different criteria in these choices leads to consideration of different approaches to design, namely the *pure performance* approach, the *joint performance/performability* approach, and the *pure performability* approach.

A *pure performance* approach corresponds to conventional design, where any augmentation is applied to the fully operational state to achieve the best improvement in performance in that state. In the *joint performance/performability* approach, the state chosen for optimisation is the one where the worst level of performance is attained, and the topology which gives the best improvement in performability is selected at the end of an iteration. The third approach is the *pure performability* approach, where both choice of the state and the topology are based on performability. In Chapter 3, the concept of a *performability index* was proposed as a means of assessing the relative significance of components in a system with respect to their failure rates, mean repair times, and degree of performance degradation caused when they fail. Its use in the choice of the state is investigated, the rationale being to apply augmentation to a state where the most important component has failed. The *randomisation technique* is used to evaluate performability, whenever required in the design approaches.

The augmentation mechanisms used are adding new links, and augmenting the capacity of existing links. From [BER87], it is suggested that optimal routing tends to avoid multiple paths. Traffic is directed along a single shortest path, as long as no overload occurs. This statement implies that the routing algorithm would favour a capacity augmentation to addition of a new link. A mathematical comparison of the two mechanisms is given, as well as an evaluation of the relative merits of these means using the networks designed. Design using both mechanisms will show that choice of the best augmentation technique depends on the scenario being dealt with, and the distribution of traffic among the links.

A performability-based analysis aims at capturing those states where components have not completely failed, but operate at a degraded level. The application of performability to design of networks has considered the network when it is fully operational, and is in a state where failure of a link causes performance to degrade. To extend the depth of the analysis, each link is assumed to be able to operate at a fraction of its capacity so that there is an additional intermediate state for each link. The effect of including intermediate states of failure is investigated.

The chapter is organised as follows. The two proposed cost functions, resulting in two descent factors are compared in Section 5.2. Execution of stages 1 and 2 of design of the 10-node network are discussed in Sections 5.3.1 and 5.3.2 respectively. The *fault-tolerance augmentation* stage is considered in Sections 5.3.3 and 5.3.4. The different sets of results are then given, where the different design approaches are compared. In Section 5.4, the networks are designed using capacity augmentation only, whereas in Section 5.5, an increase in connectivity is used as sole augmentation mechanism in the design of networks. The design approaches are compared in terms of the performability of the networks as the iterations proceed in Section 5.6. In Chapter 2, the approach adopted by Tan and Kang [KAN92] to fault-tolerant design was described. It is compared with a performability-based approach in Section 5.7. In Section 5.8, the way in which performability-based approaches, which optimise performance in states of failure, affect improvement in performance in the fully operational state is considered. In Section 5.9, conclusions from a comparison of the various design approaches are discussed. Sections 5.10 and 5.11 are devoted to a comparison between the two augmentation mechanisms using mathematical arguments in the first section and evaluation of networks designed in the latter one. Both mechanisms are combined to generate results presented in Section 5.12. The design of networks, when intermediate states of failures are included, is considered in Section 5.13, and conclusions based on results generated are given in Section 5.14.

5.2 Comparison between use of the two cost functions in the descent factor

In Chapter 4, the first definition of the cost of a network was expressed in terms of the cost of its nodes and links. During a given observation period, components of a network may fail and require different lengths of time to be brought back to operation. Repair is associated with a certain cost, and it is only reasonable to estimate the amount of investment that is expected to go into maintenance. The second definition, therefore, consists of the cost of the network in terms of its components, and the average repair cost expected to be incurred over a period of time. A 6-node network, where the nodes are configured as shown in Figure 5.1, is used to illustrate the use of the two cost functions.

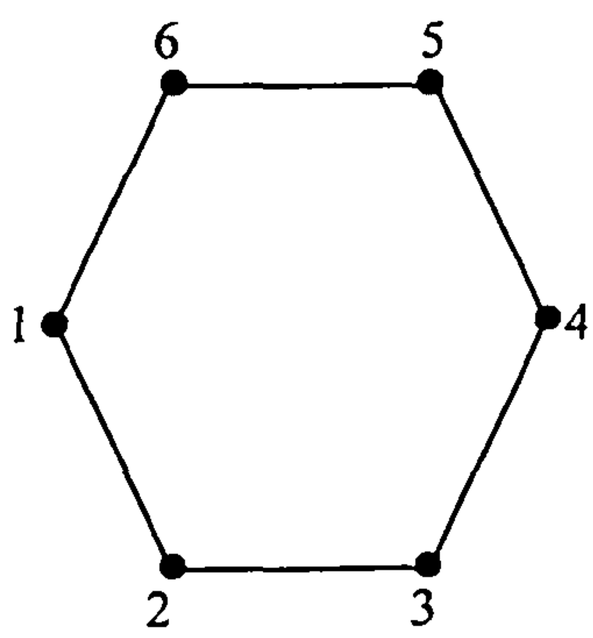


Figure 5.1 - 6-node ring network

Two unidirectional links connect every pair of nodes, but failure of one unidirectional link is treated as total failure of direct communication between the connecting pair of nodes. The links are associated with failure rates which are assumed to range from 1 to 5 per 100 days. The mean repair time varies between 24 and 25 hours. Each link is further associated with an average repair cost which varies between 100 and 300 units for the existing links.

The GPM-based routing algorithm is applied to the above network in all of its states, as dictated by single failures of its links. The worst performance is obtained when links connecting nodes 5 and 6 fail. Based on the distribution of load over the remaining links, the suggested augmentation mechanisms are addition of links between nodes 1 and 4

giving a network that will be referred to as network1, or the addition of links between nodes 2 and 4 giving a network referred to as network2. Assume that both potential links have the same failure rates, and mean repair times. However, the links between nodes 1 and 4 are associated with an average repair cost of 600 units whereas the ones between nodes 2 and 4 have a mean repair cost of 75 units. These two options will be investigated. The performability distributions of the initial network and the potential new topologies are evaluated using the *randomisation* technique. From a performability point of view, the addition of links between nodes 1 and 4 is more beneficial. However, setting up the network is more expensive. Data pertaining to performability and cost of the network is summarised in Table 5.1. Performability is taken at the point where reward equals 0.8. The average repair cost of the network is evaluated using the algorithm by Iyer *et al.* [IYE86], described in Chapter 3. The total cost of the network includes the cost of setting up the network and average repair cost over a time period of 10 days.

Network	**P[$ACR(10) \leq 0.8$]**	**Cost of setting up the network**	**Average repair cost of network**	**Total cost of network**
Initial	0.3171	3001.65	282.22	3283.87
Network1	0.1745	3253.65	480.08	3733.73
Network2	0.2471	3203.38	298.47	3501.85

Table 5.1 - Cost and performability of network

As explained in Section 4.3, the performability descent factor is evaluated as $\frac{\Delta \text{performability}}{\Delta \text{cost}}$. In the fault-tolerance augmentation stage, the mechanism is sought which gives the maximum decrease in performability at a minimum increase in cost.

Based on the data in Table 5.1, evaluation of descent factor 1, which excludes cost of repair is as given below.

For network1, Descent Factor 1 = -0.000566

For network2, Descent Factor 1 = -0.000035

A comparison of the descent factors leads to choice of network1. The higher cost of setting up the network is offset by the improvement in performability.

Consider evaluation of descent factor 2, which includes cost of repair.

For network1, Descent Factor 2 = -3.17 x 10e-4

For network2, Descent Factor 2 = -3.21 x 10e-4

When the average cost of repair of the links is taken into consideration, network2 is preferred because the investment required by network1, in terms of setting up the network and maintaining it for a period of time, is not justified by the degree of improvement in performability.

This simple example illustrates how a trade-off needs to be achieved between the objective of attaining the best performability measure and the constraint imposed by a budget. Given that the fault-tolerance needs to be evaluated over a period of time, it is meaningful to include the average cost of repair expected to be incurred, when estimating the financial requirements of the network. In the example considered, although the two networks behave in a similar way in respect of component failure and repair, network1 is more costly to maintain. Therefore, in a practical situation, where costs act as a major constraint, an estimate of the amount of money required for repair is desirable, in addition to the initial cost of setting up the network.

However, an implementation of the algorithm by Iyer *et al.* in Matlab was found to give unreasonable results as the size of the network grows; especially if the difference in mean repair times is small. This was due to manipulation of very small numbers. Therefore, although, average repair cost has been shown to be an important concept, it is not used in derivation of results in the following sections, when cost needs to be included.

5.3 Application of the proposed design technique

The capability of the proposed design method is illustrated with the design of a 10-node network. The steps involved in the whole process are discussed in the following sections.

5.3.1 Execution of Stage 1: initial topology generation stage

The locations of the nodes constituting the network to be designed are shown in Figure 5.2. In this first stage, the minimal spanning tree that connects the nodes has to be generated. In derivation of this initial configuration, the weights associated with the links may depend on the physical distance, or the traffic demands, between the nodes. The topology shown in Figure 5.2 corresponds to the minimal spanning tree designed based on a distance criterion whereas a traffic demand criterion is used in generation of the network in Figure 5.3. In these figures, a single line represents the two unidirectional links connecting each pair of nodes.

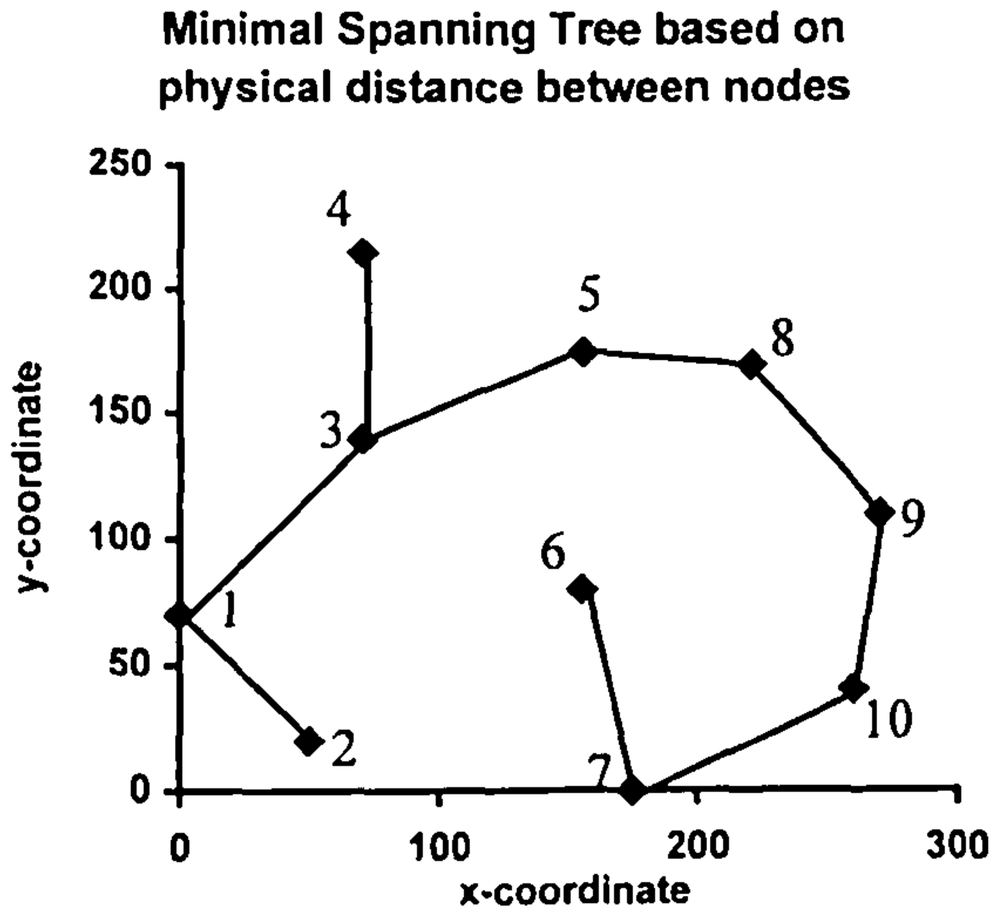


Figure 5.2 - Minimal Spanning Tree based on physical distance

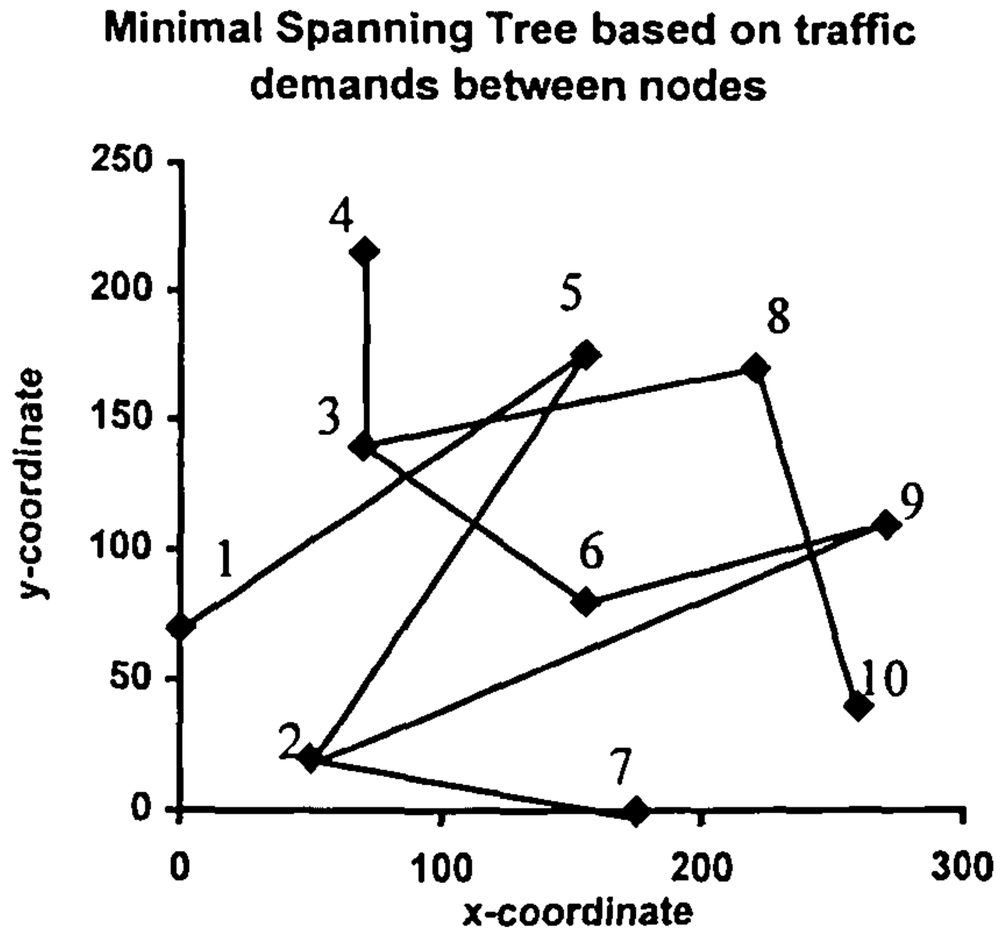


Figure 5.3 - Minimal Spanning Tree based on traffic demands

5.3.2 Execution of Stage 2: connectivity establishment stage

In the above networks, failure of any link leads to disconnection. The *connectivity establishment* stage is required to ensure that at least a 2-connectivity criterion holds. Here, again, addition of more links can be based on distance, or demands, depending on the factor used in design of the network at Stage 1. The topologies generated after execution of the second stage using distance, and traffic demands, as criterion are shown in Figures 5.4 and 5.5 respectively.

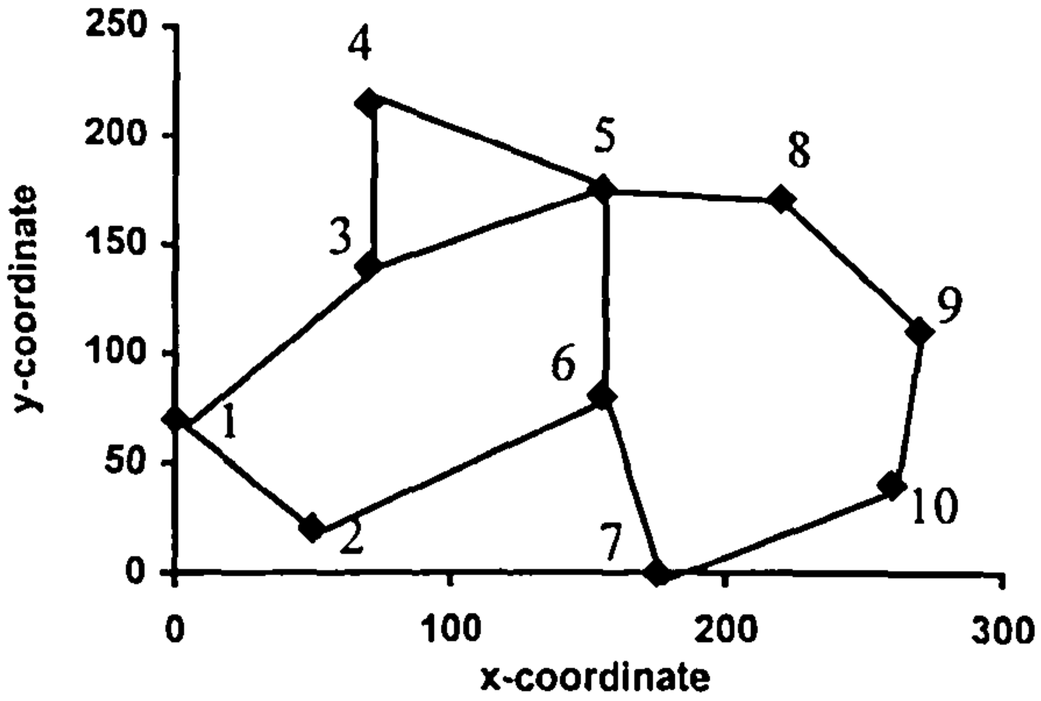


Figure 5.4 - Network designed using a distance criterion at Stage 2

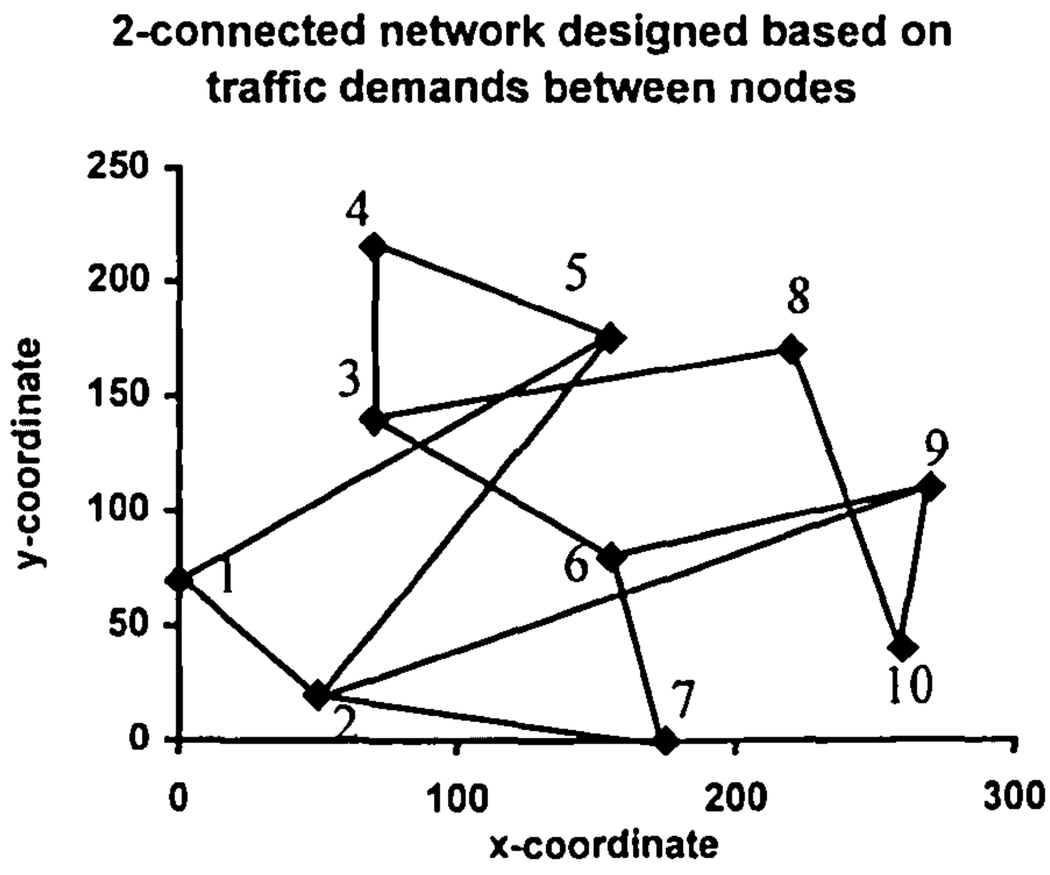


Figure 5.5 - Network designed based on traffic demands at Stage 2

Given the above networks, their performability distributions can be derived. This requires evaluation of the performance of the network in all of its failure states, as dictated by failures of its links. Since the network is minimally 2-connected, there is always a route for packets to be transmitted from their source to their destination when a single link failure occurs. Therefore, it is assumed, given that only single link failures are allowed, that no packets are discarded.

A comparison of the two networks shows that although the topology derived on the basis of the distance criterion has a higher overall capacity, it is actually cheaper. This is due to the fact that cost of a link is a function of its length. In terms of performance, the network designed using a demand criterion tends to be better in some states of failures than the topology derived based on distance. This is not always the case though, and when comparing the performance levels attained by both networks, the worst level of performance is obtained when the links between nodes 3 and 8 fail in the topology shown in Figure 5.5.

Data pertaining to the reliability of the components is as follows. The links are assumed to be associated with failure rates which vary between 1 and 5 per 100 days, and the average repair time of a link is set to be 24 hours. Given the performance and reliability data, the performability model can be derived, and performability is evaluated as a distribution of average accumulated reward over a period of 10 days. A comparison of the

performability distribution of the two networks is given in Figure 5.6, where rewards are assigned taking into consideration the range of performance levels being covered. The distributions obtained for the networks in Figures 5.4 and 5.5 are labelled *distance* and *demand* respectively.

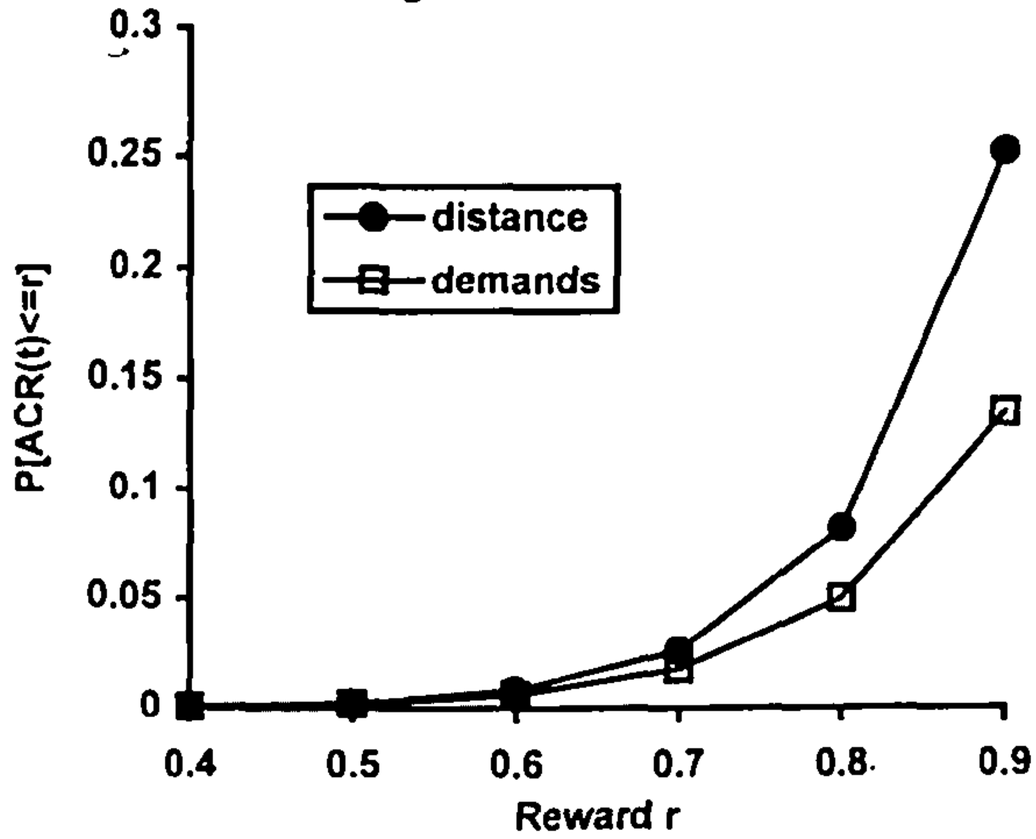


Figure 5.6 - Performability distributions of networks designed using distance and traffic demands as criterion

The fact that performance levels tend to be better for the network designed based on demands is reflected by its performability profile being closer to the ideal. The network appears to be more fault-tolerant, even though it is the network that delivers the worst performance level in one of its states.

5.3.3 Execution of Stage 3: fault-tolerance augmentation stage

The *fault-tolerance augmentation* stage is the most important stage, where, given a network as input, augmentation mechanisms are applied to improve the network's ability to withstand failures.

As explained in Section 4.2.5, in an iteration of this stage, a state needs to be chosen at the beginning of an iteration, and the best topology is selected at the end of the iteration. The different criteria on which these choices can be based generate different combinations

of choice of state / choice of topology that can be used in design. In the following sections, three different approaches will be investigated and compared. These are the *pure performance* approach, the *pure performability* approach and the *joint performance / performability* approach.

In a *pure performance* approach, the state chosen for optimisation is the fully operational state, and the topology chosen at the end of an iteration is the one which performs best in that state. States of failure are not considered.

In contrast, in the performability-based approaches, assessment of performance is not confined to the fully operational state, but also takes into consideration performance in states of failure. For both pure performability and joint performance/performability-based designs, the choice of the topology at the end of an iteration is based on performability. In other words, the means that leads to best improvement in performability, hence fault-tolerance is chosen. The difference between the two approaches is in the choice of state. In a pure performability approach, the concept of *performability index* is used to select the state where augmentation is to be investigated whereas in a joint performance/performability approach, the selection of the state is based on performance only, that is, the state of worst performance is chosen.

In generation of most results for comparison, the mapping of performance, measured in terms of average number of data units in the network, on to rewards is done according to *definition 1* as follows:

For each performance level x,

$if\ x < 150,\ reward = 1.0,$

$if\ 150 \leq x < 250,\ reward = 0.5\ and$

$if\ 250 \leq x,\ reward = 0.0.$

If comparison is required between two topologies associated with the same reward structure, the range of performance levels covered can be taken into consideration and

definition 2 may be used. The two definitions were discussed in Section 4.4. It should be recalled that when designing with respect to performability, the ideal distribution of average accumulated reward which can be aimed at is a step function at $\mathbf{u}(r - 1.0)$.

In the first set of results, the effect of subjecting the networks of Figures 5.4 and 5.5 to *fault-tolerance augmentation* is investigated.

5.3.4 Application of fault-tolerance augmentation

In this section, fault-tolerance augmentation is applied with regard to the initial networks shown in Figures 5.4 and 5.5. Although the augmentation mechanisms, adding new links and augmenting the capacity of existing links, can be combined, results are obtained when either one of the techniques is used. The cost of the network is not taken into account in derivation of these results. Figures 5.7 and 5.8 show how performability taken at the point where reward equals 0.8, that is $P[ACR(t) \leq 0.8]$, varies as the iterations of design proceed; with t being taken to be 10 days.

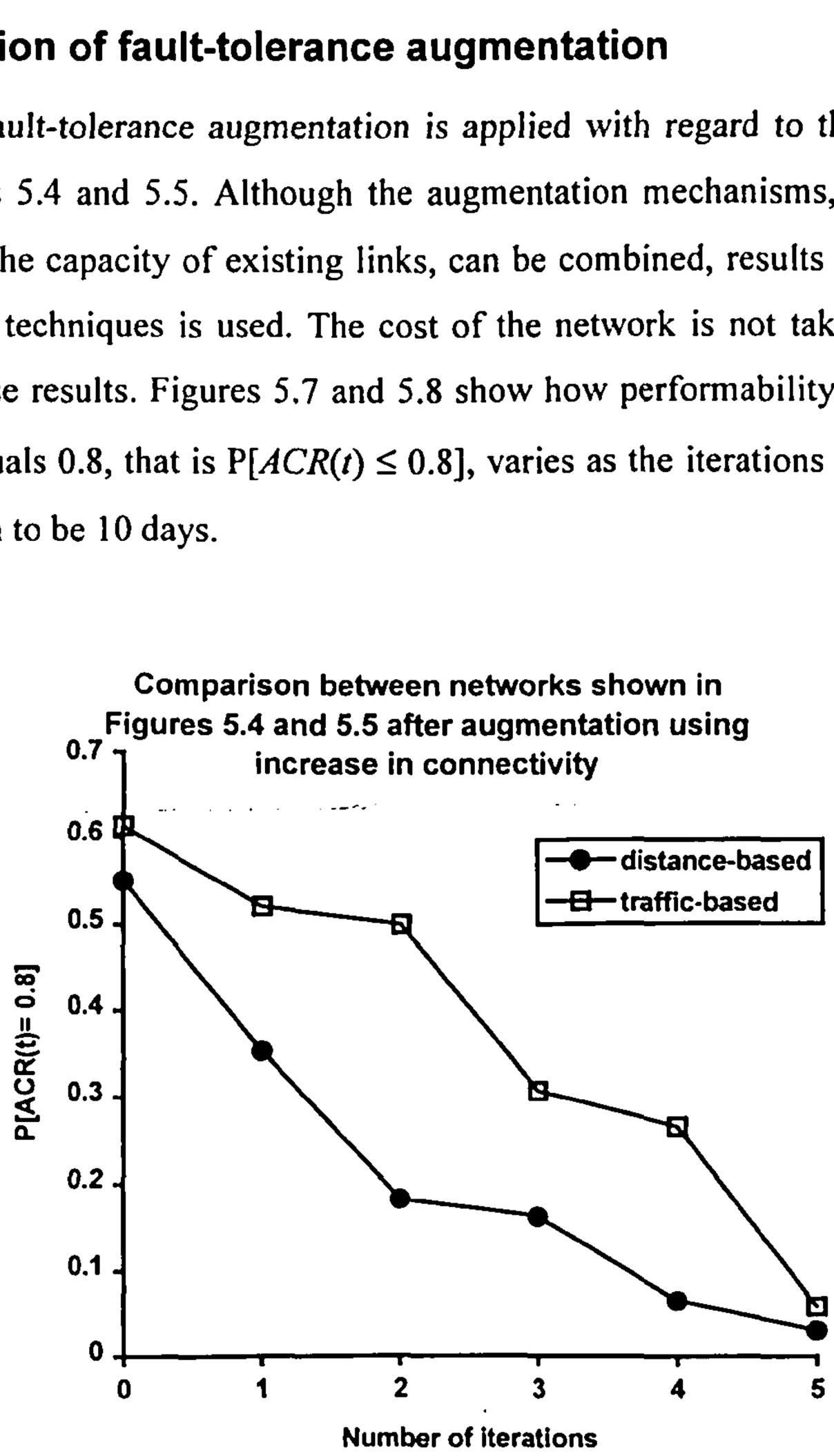


Figure 5.7 - Point performability per iteration of design of networks initially based on distance and demands, using increase in connectivity

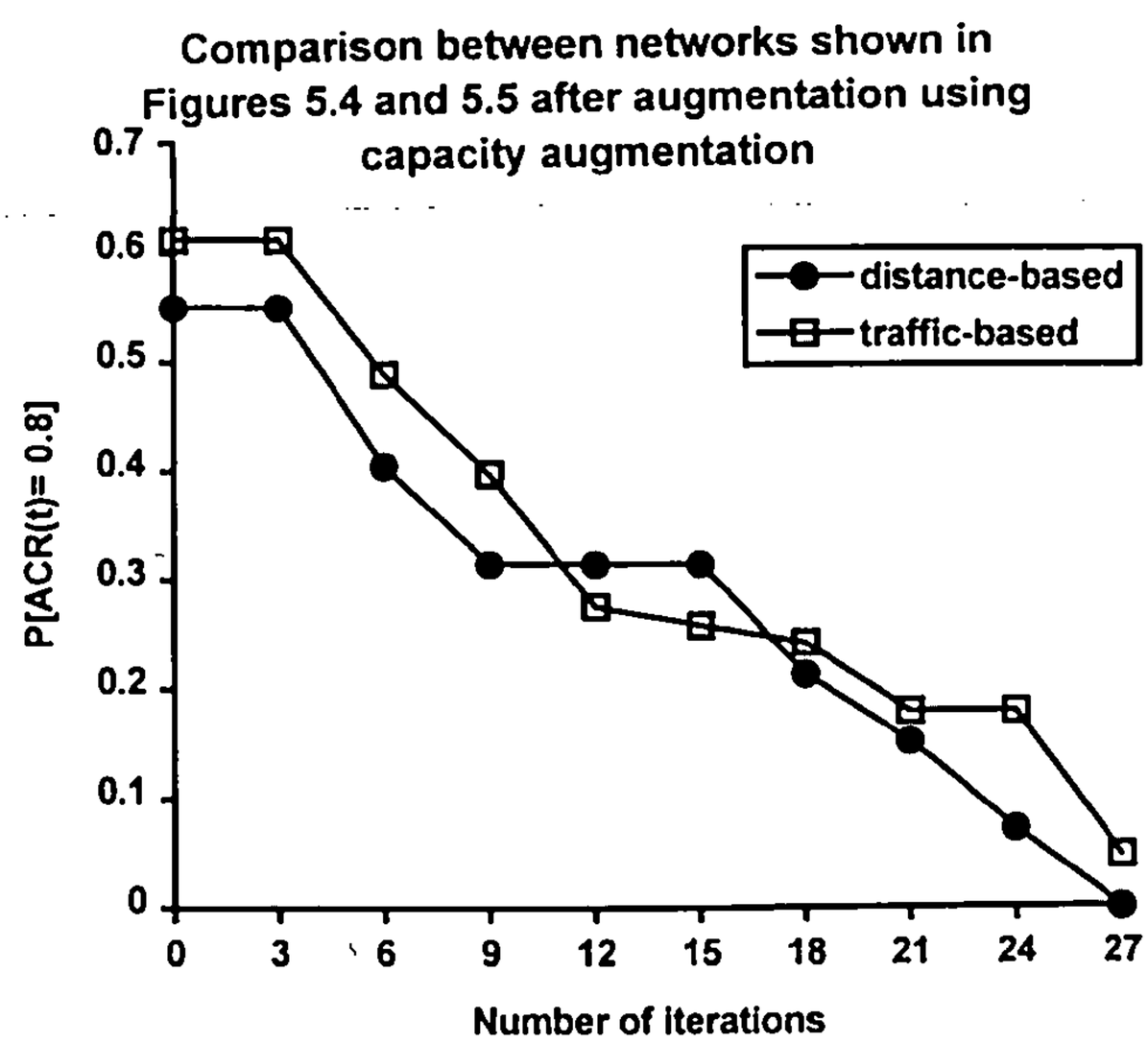


Figure 5.8 - Point performability per iteration of design of networks initially based on distance and demands, using capacity augmentation

Generally, performance levels were found to be better for the initial network designed using a traffic-based criterion. However, according to the first definition of rewards, an increase in connectivity appears to generate a better network faster when the initial network design is based on distance. This is because the addition of links causes more improvement in the various levels of performance in this case. When capacity augmentation is used, this is not always true, as shown in Figure 5.8.

Augmentation of the network designed according to a distance criterion will be used for discussion of further results in the sections which follow.

5.4 Set A - Capacity augmentation based design

In this set, capacity augmentation is used as sole fault-tolerance augmentation mechanism. Given the network at the beginning of an iteration, a state is chosen where optimisation is to be applied. The most loaded links are identified, and the effect of an increase in the capacity of each of them is investigated individually, giving rise to a number of topologies. The best topology is chosen. The iterations proceed until a suitable criterion is met.

The three approaches as dictated by combinations of choice of state and choice of topology are investigated. Comparison between these different combinations is carried out at a stage where performance in the fully operational state is less than 20, which is the stopping criterion for the pure performance approach. This does not necessarily mean that the objective performability profile has been reached when design is performability-based. Performance is being measured as the average number of data units in the network.

In Table 5.2, the performance levels obtained for the network operating in all the allowable configurations are given for four different networks. Adopting the notation used in previous tables, links (*i-j*) represent the unidirectional links connecting nodes *i* and *j*, and their failure results in there being no direct means of communication between the nodes in the resulting topology. The initial network corresponds to the topology shown in Figure 5.4. The three other networks represent those obtained after subjecting the initial network to fault-tolerance augmentation according to the pure performance, joint performance / performability, and pure performability-based approaches.

Configuration	Performance of	Performance of network designed according to approach based on		
	Initial network	Pure Performance	Performance/ Performability	Pure Performability
Fully operational	56.56	19.83	19.67	19.82
Links (1-2) fail	1290657.00	1178432.00	25.09	25.10
Links (1-3) fail	1.21e7	1.06e7	39.74	40.16
Links (2-6) fail	944.89	21.69	259.78	33.81
Links (3-4) fail	1477054.00	1477022.00	41.72	41.46
Links (3-5) fail	8722924.00	8630677.00	45.83	46.45
Links (4-5) fail	478.94	19.87	232.28	441.15
Links (5-6) fail	105.92	38.16	28.59	28.30
Links (5-8) fail	1.27e7	5333499.00	136.52	136.84
Links (6-7) fail	113508.90	25.39	89.85	89.17
Links (7-10) fail	1880191.00	25480.16	51.98	49.10
Links (8-9) fail	8735316.00	2941809.00	38.64	38.22
Links (9-10) fail	7971638.00	1672424.00	47.59	46.16

Table 5.2 - Performance levels of the networks designed when capacity augmentation is used

Note that performance is given by $\frac{F_{ij}}{C_{ij} - F_{ij}}$ which is the number of data units in the network.

The performability distributions of the networks compared in Table 5.2 are plotted in Figure 5.9, as a distribution of average accumulated reward for a period of 10 days. The performability of the initial network is shown by the line labelled *initial*. The remaining labels used in Figure 5.9 represent the type of approach used in design.

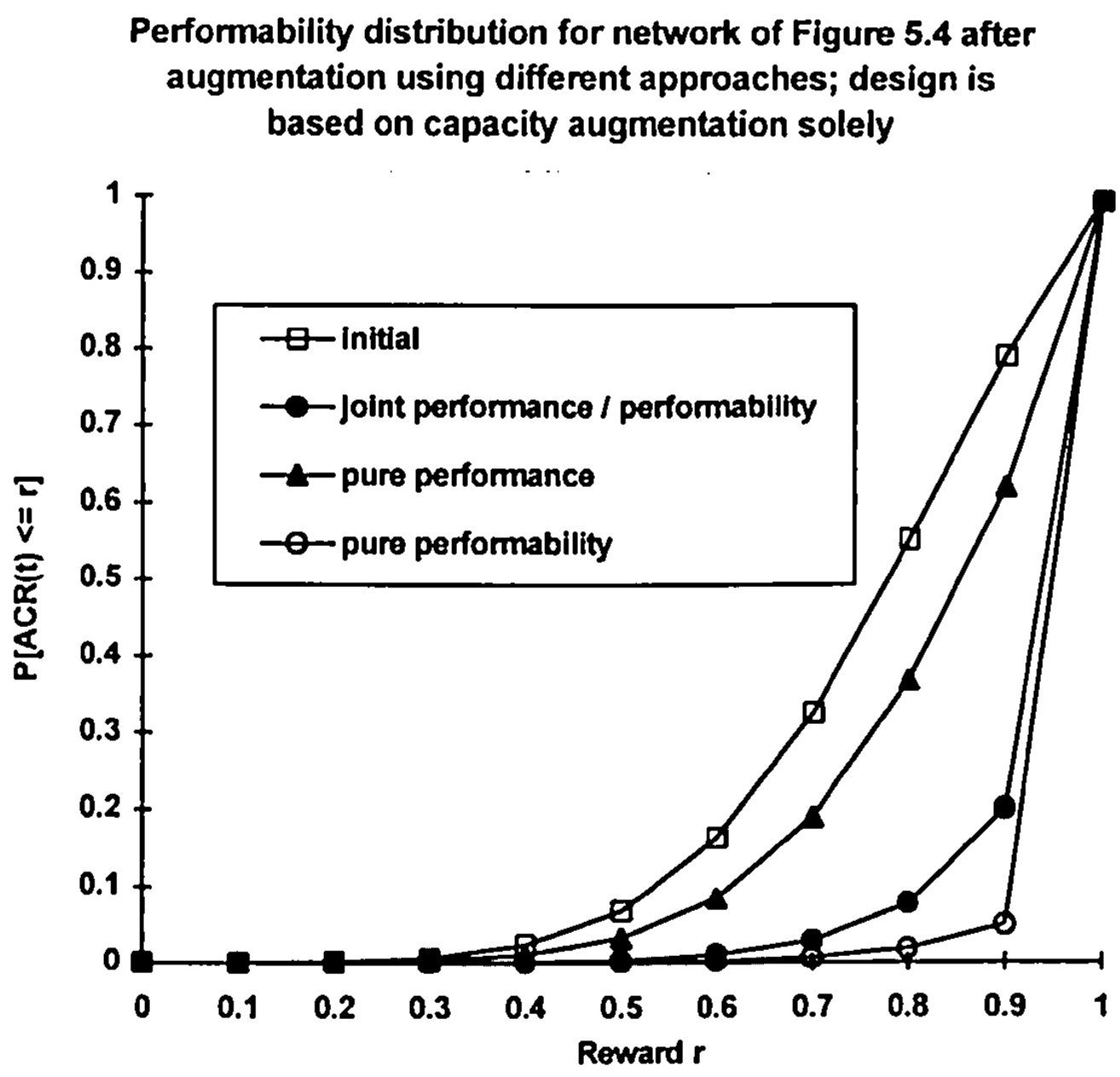


Figure 5.9 - Performability distributions of the networks designed using capacity augmentation

From Table 5.2, the performance of the initial network in some states of failure is highly degraded as compared with the level obtained in a fully operational state. On augmentation according to the pure performance approach, performance is optimised in the fully operational state, but with no regard for performance in states of failure. Given that the capacity of the overall network is increased, performance is enhanced to various degrees in the states of failure.

Application of a performability-based design aims at improving performance in all the possible network states. As shown in Table 5.2, at a stage where performance meets the requirement of being less than 20 in the fully operational state, performance in the failure states is considerably improved when choice of the topology is performability-based. This translates into the distribution of average accumulated reward for networks designed using performability-based methods being closer to the ideal step function, than the one generated by a pure performance approach. This can be seen from Figure 5.9. In this specific set of results, using the performability index to choose the state for optimisation appears to

produce a better network than when using performance to do so. Furthermore, in the performability-based design, both the pure performance and joint performance / performability methods, 25 augmentations were sufficient to meet the requirements whereas 34 iterations were necessary in the pure performance approach.

5.5 Set B - Increase in connectivity based design

In this set, fault-tolerance augmentation is achieved through increases in connectivity only. Once the state where optimisation is to be applied is selected, the load on the links can be determined. Using the rules described in Section 4.2.5.4, the different possibilities as to where to add potential links are identified, yielding a number of topologies. The best topology can then be chosen with respect to the design criterion being used. Iterations proceed until requirements are met. The network of Figure 5.4 is used as initial network.

The pure performance, joint performance/ performability, and pure performability-based methods are investigated. Comparison is carried out at a stage where performance in the fully operational state is less than 30. In Figures 5.10 and 5.11, the results are shown in terms of a distribution of average accumulated reward for a time period of 10 days. The networks being compared are obtained after 5 iterations for the pure performability approach, 6 iterations for the joint performance / performability approach, and 8 iterations when design is based on a pure performance criterion.

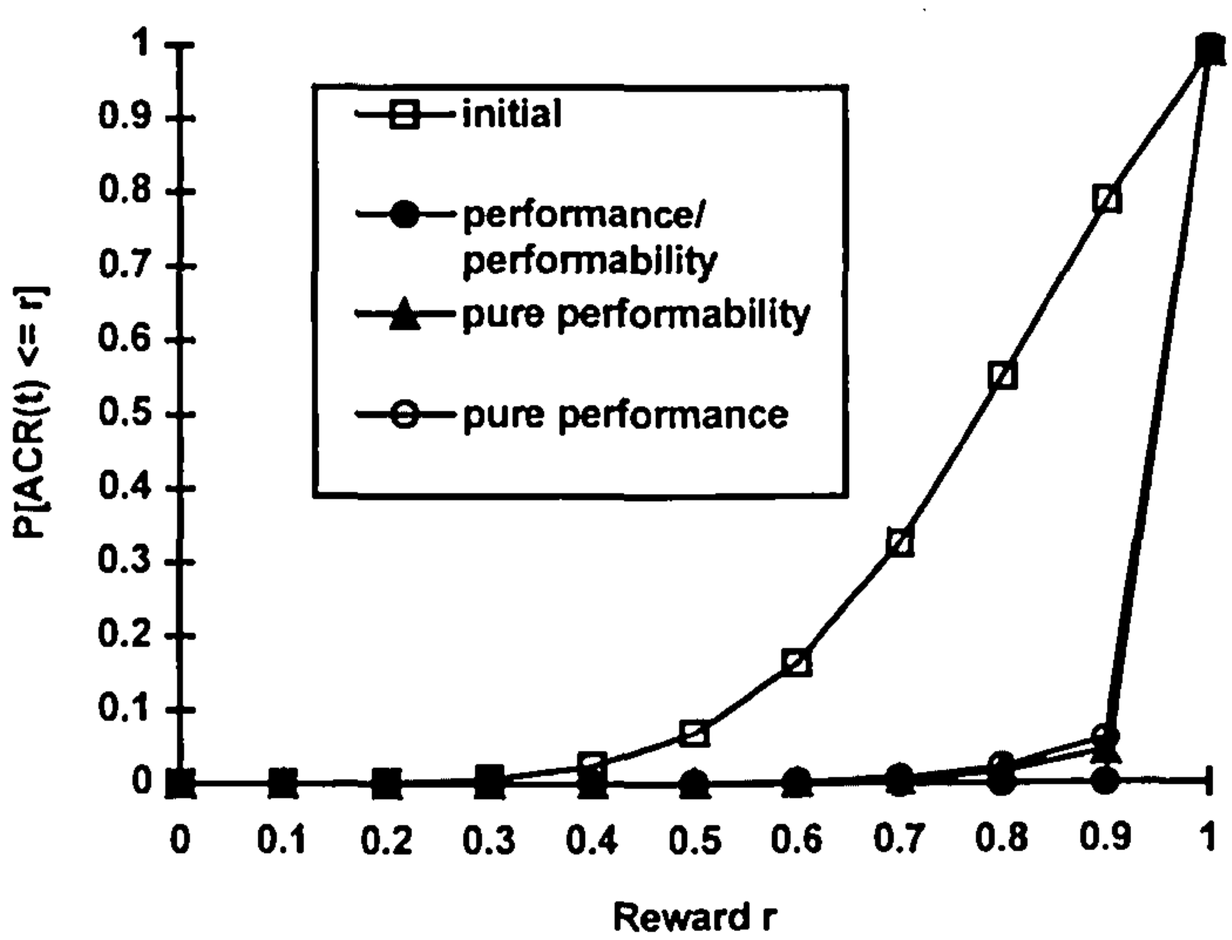


Figure 5.10 - Performability distributions of the networks designed using increase in connectivity

From Figure 5.10, adding links according to all three approaches causes a considerable improvement in the distribution of average accumulated reward. This is due to performance being enhanced in all the states in which the network can be. In Figure 5.11, a closer look is taken at the different approaches to fault-tolerance augmentation.

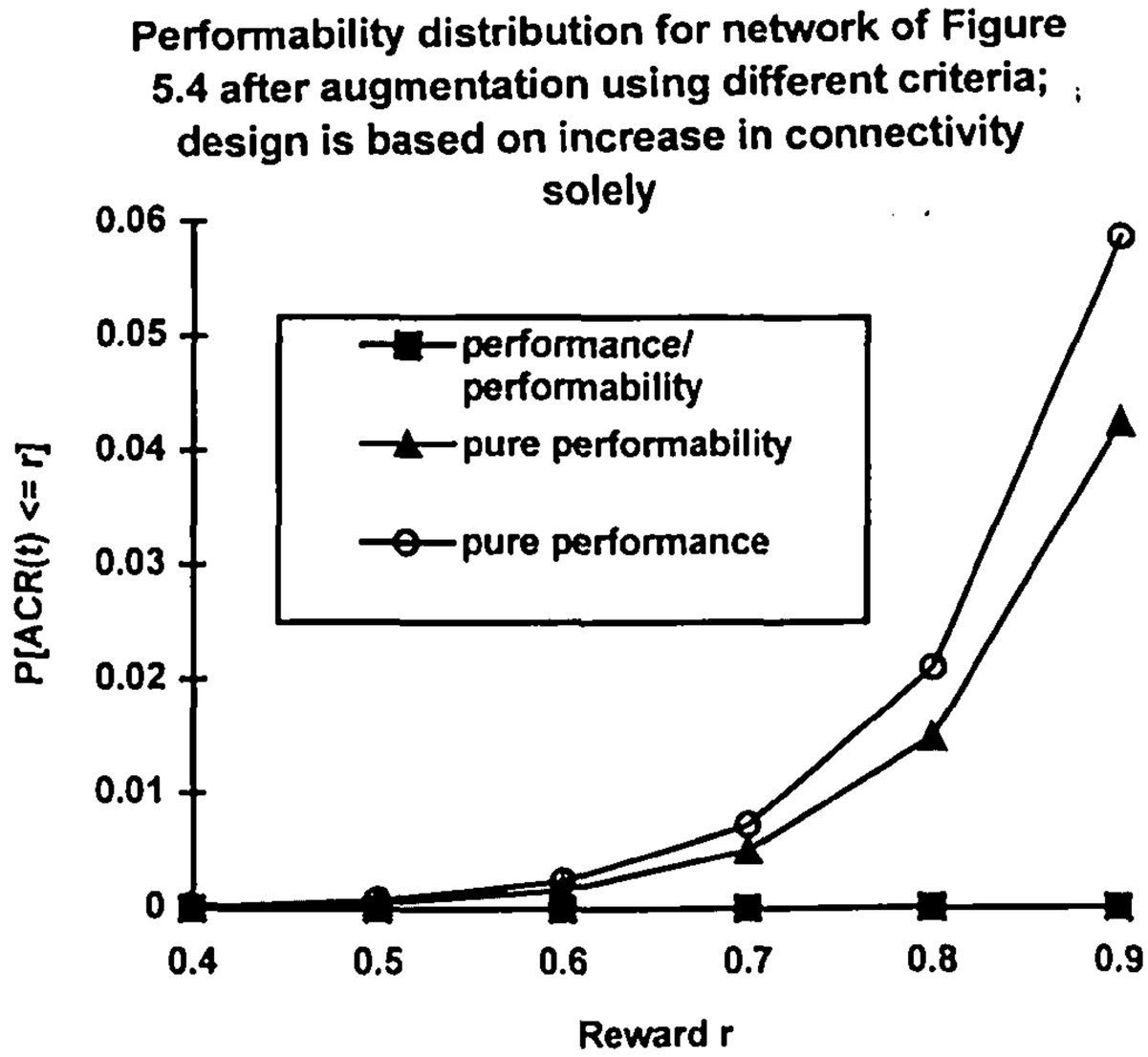


Figure 5.11 - Performability distributions of the networks designed using increase in connectivity over reward range [0.4,0.9]

From Figure 5.11, performability-based design methods do yield networks whose distribution of average accumulated reward is closer to the ideal step function. Addition of links according to a pure performance criterion contributes to enhancing performance in all the states, but the performability-based methods are better. Furthermore, in this specific example, networks that are more fault-tolerant are obtained on the addition of fewer links than is required if design is carried out according to a pure performance method. A joint performance / performability approach generates the best network. Its distribution of average accumulated reward is coincident with the ideal step function.

In Figure 5.12, the average distribution of average accumulated reward is plotted for the networks generated after 5 iterations of the design method. The performability-based methods outperform the pure performance approach, but at this stage, the pure performability-based approach seems to give a better network than the joint performance / performability-based method. This is in contradiction with the results shown in Figure 5.11, where a joint performance / performability method generates the best final network. The

way in which performability varies at each iteration for the different methods is investigated in Section 5.6.

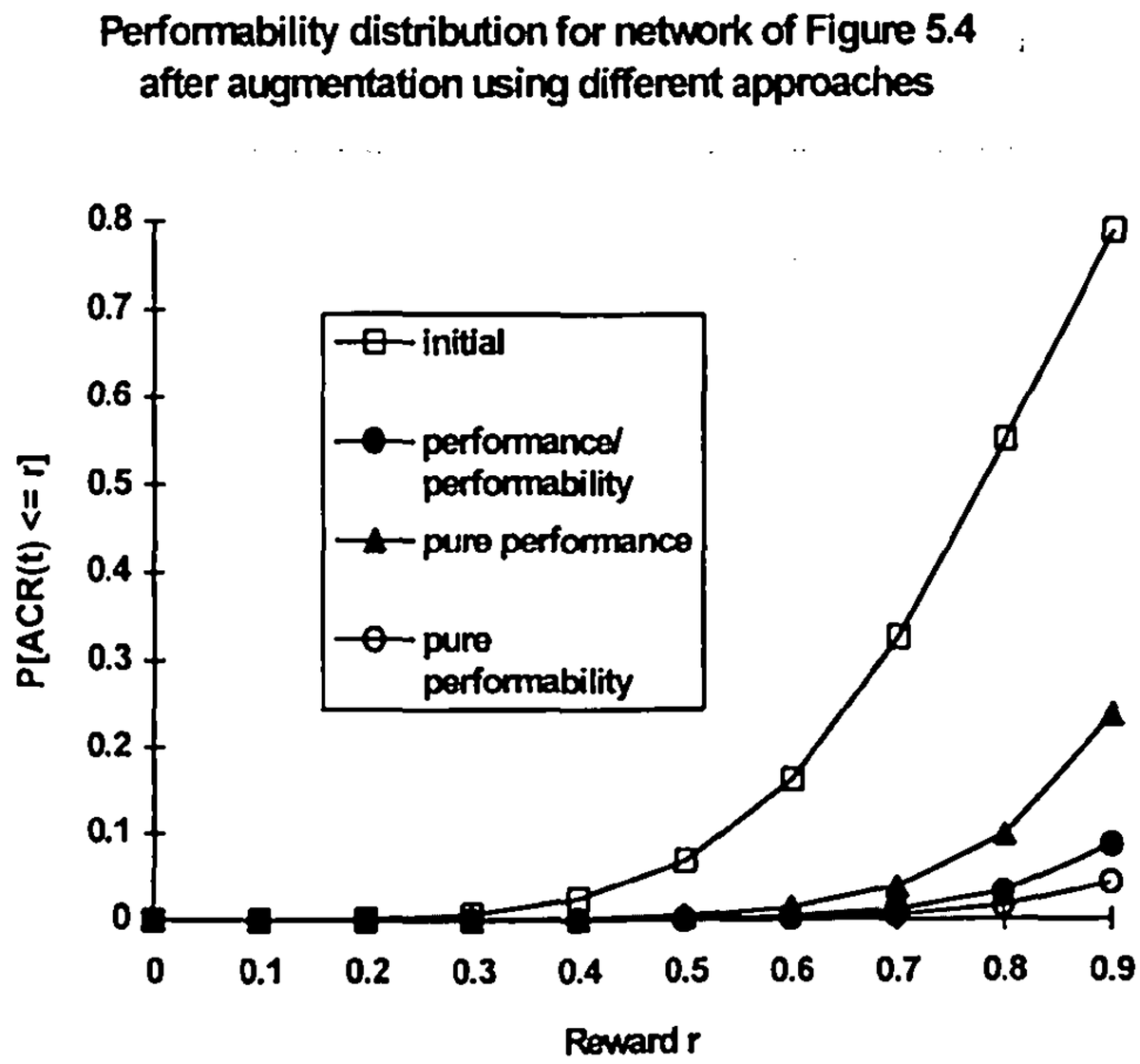


Figure 5.12 - Performability distributions of the networks after 5 iterations of an increase in connectivity-based approach

5.6 Set C - Comparison at different numbers of iterations

In this section, the effects of applying different fault-tolerance mechanisms to the network in Figure 5.4 are observed after each iteration of the design. The three design approaches are compared. Performability is now taken at the point where reward equals 0.8, and is therefore given by P[$ACR(t) \leq 0.8$], where time t is set to be 10 days. Figures 5.13 and 5.14 show how this point performability varies after different numbers of iterations for all three approaches. In Figure 5.13, results are obtained when design is based on capacity augmentation only, and are plotted after every three iterations of design. Increase in connectivity is used as sole augmentation mechanism in generation of results for Figure 5.14.

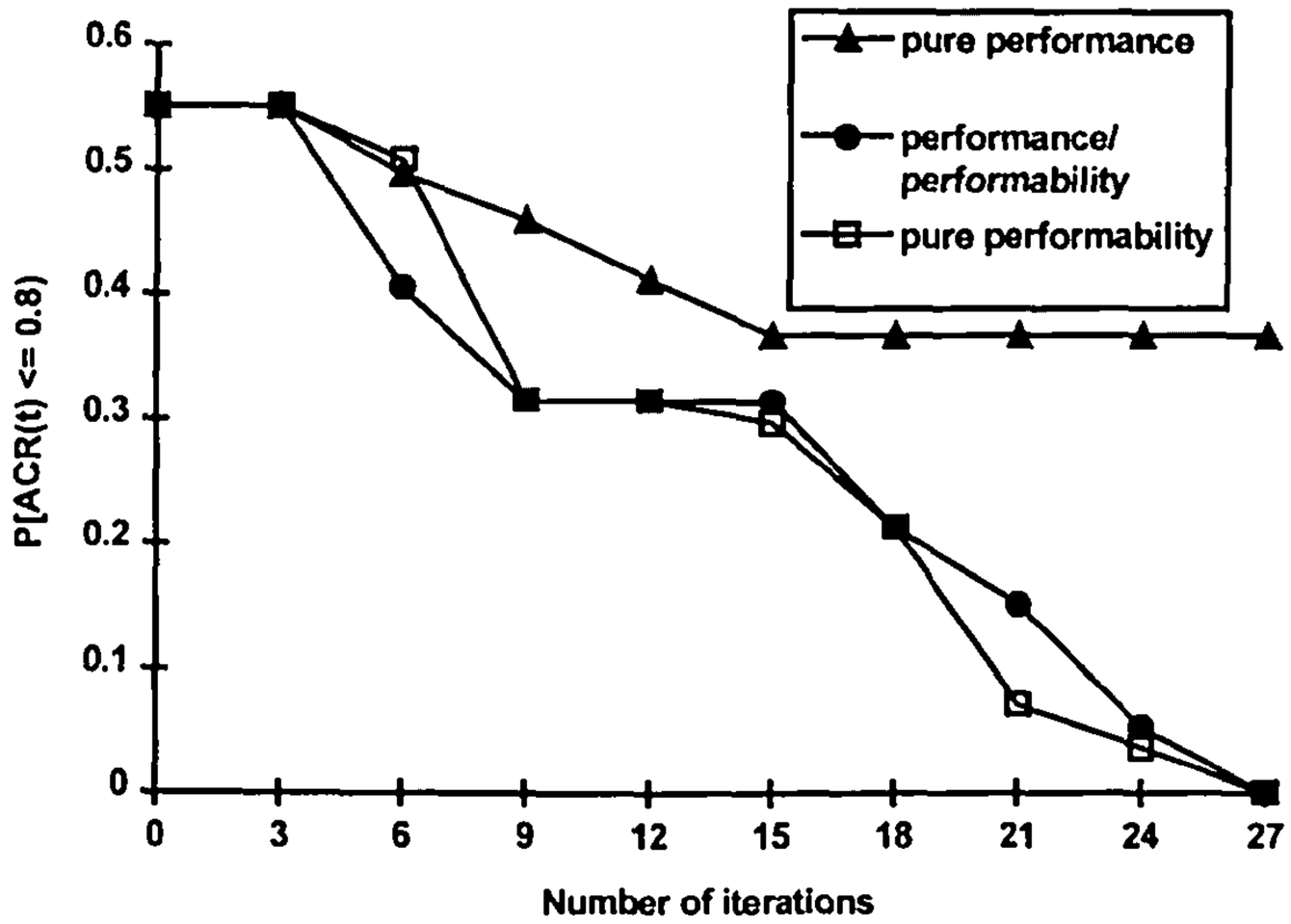


Figure 5.13 - Point-performabilities of networks designed using capacity augmentation after every three iterations

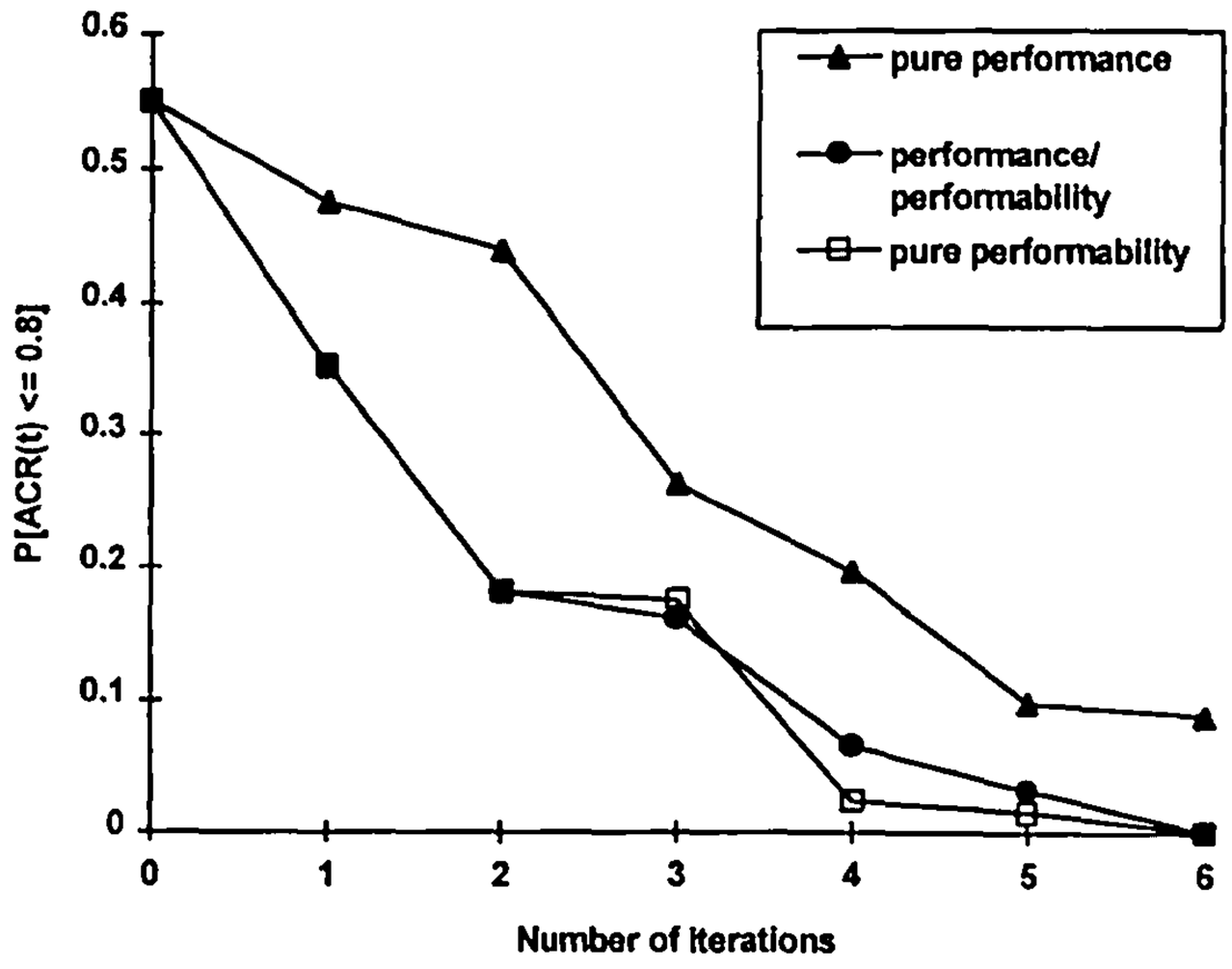


Figure 5.14 - Point-performabilities of networks designed using increase in connectivity at each iteration

In the results derived for Figure 5.14, the cost of the network was not taken into consideration. Including cost in the descent factor used in the fault-tolerance augmentation stage gives the results shown in Figure 5.15. Given that all the potential links are assumed to be associated with equal failure rates, and mean repair times, and require the same cost of repair, the overall average cost of repair is not taken into consideration; the descent factor used is given by the first definition in Section 4.5.1.

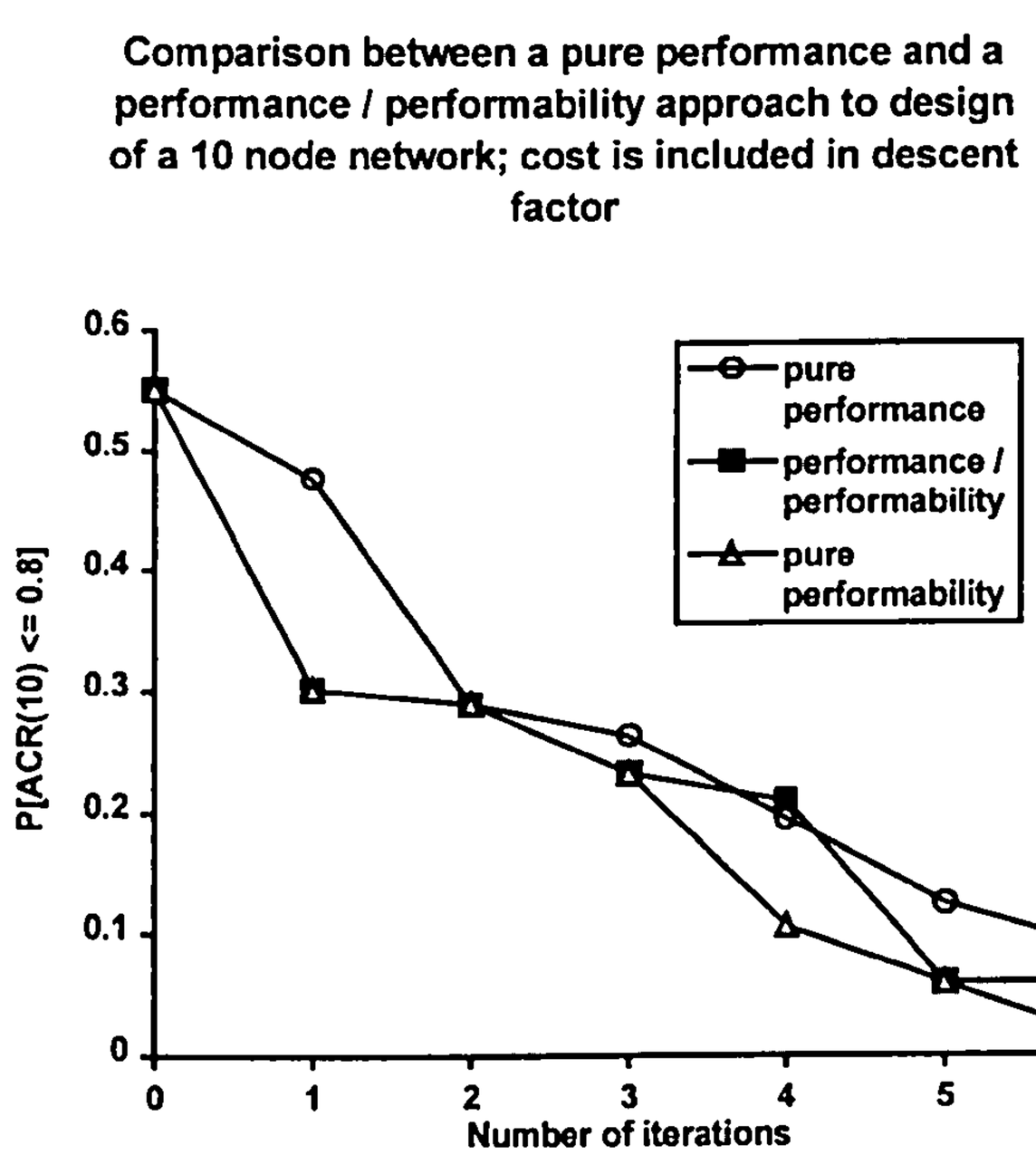


Figure 5.15 - Point-performabilities of networks designed using increase in connectivity when cost is considered

When cost is not taken into account, the performability-based approaches outperform the pure performance approach, by giving lower point-performabilities as the iterations proceed. The ideal value for point-performability at any reward not equal to 1, is zero. A comparison between the pure performability and joint performance/performability approaches is not conclusive in the sense that there is no indication as to which approach produces a better network faster. If cost is included in the descent factor, the advantage of using a performability-based design is less apparent. This is due to the fact that cost acts as a constraint and may hinder the choice of the best link with respect to performability.

In this specific example, the performability-based methods yield networks that are more fault-tolerant after fewer iterations than in the case when the pure performance approach is used. The implication is that a better network can be obtained using less resources if design is based on performability. The network is better in terms of its ability to withstand failures, that is, in terms of its performance in its failure states. This does not necessarily mean that performance in the fully operational state is better, but it is at an acceptable level. The reason for this is that by optimising in a state of failure, and by basing the choice of the topology on performability, resources are being used in such a way as to improve the average performance over all the allowable states. It is possible that in so doing, resources are better positioned so that fewer additional links, (or less extra capacity), are required to achieve the acceptable performance levels in all the states. From an economic viewpoint, the implication is that, depending on the locations of the additional resources, a more fault-tolerant network can be obtained at a lower cost.

5.7 Set D - Comparison between Tan and Kang's approach and a joint performance/ performability - based approach

Tan and Kang [KAN92] have proposed an approach to design fault-tolerant networks using capacity augmentation. The technique was reviewed in Chapter 2. In essence, they apply optimisation in the state of failure associated with the worst performance, but do so with respect to performance in that particular state only. In other words, the topology chosen at the end of an iteration corresponds to the one which gives best improvement in performance in that state of failure. The way in which a capacity augmentation affects performance in other states of failures is ignored; and furthermore, aspects of reliability, relating to the failure rates of the components, and average time necessary for repair, are not taken into consideration. In effect, this approach corresponds to one where both choice of state and topology are based on performance.

Adopting this approach in the *fault-tolerance augmentation* stage is now investigated; and the process is compared with the joint performance / performability approach when both are applied. The initial network topology is taken to be that shown in

Figure 5.4. The difference between the two schemes lies in the criterion used to select the topology at the end of an iteration.

Results show that, in this specific case, the networks generated by the two approaches can converge after a certain number of iterations. In other words, after x iterations, the same networks are obtained. For example, after 5 iterations, both methods give the same network. The same five capacity augmentations have been applied, but not in the same order. Figures 5.16(a), (b), (c), and (d) compare the networks generated after execution of each iteration from iterations 1 to 4. Given that at the 5th iteration, the same network is obtained by both methods, their performability distributions coincide and are not shown. The range of rewards is restricted to [0.6, 0.9] so that the distributions can be more clearly distinguished.

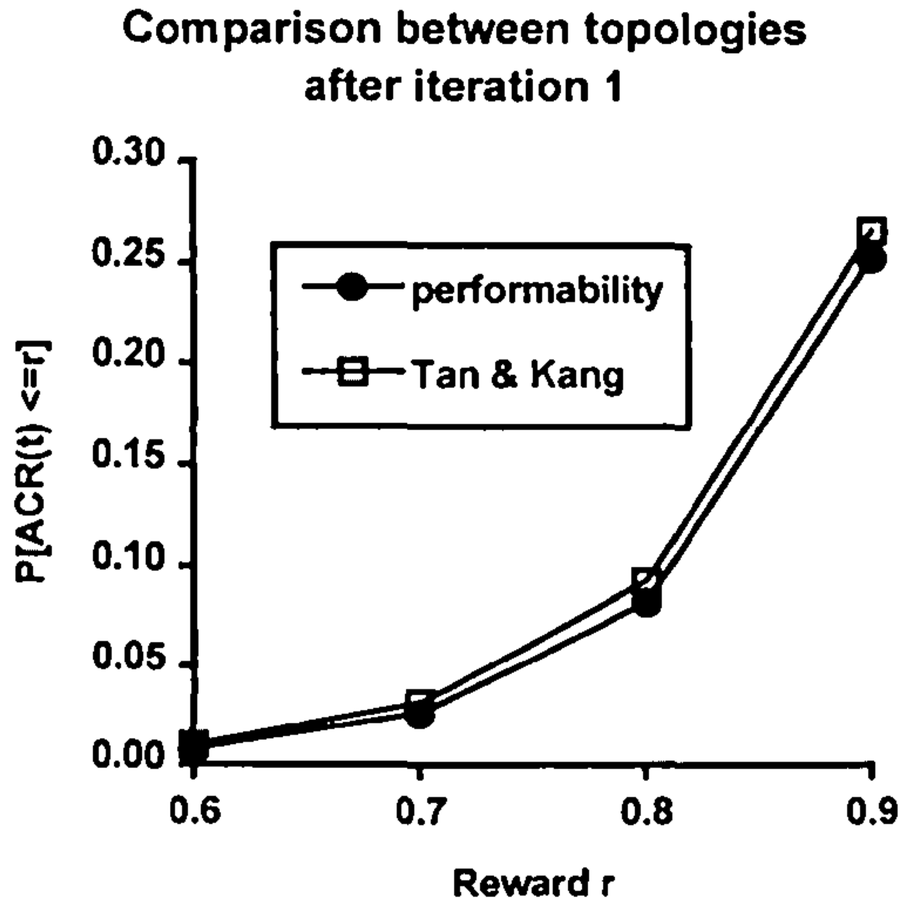


Figure 5.16(a) - Comparison between Kang's and the performability-based approach after the 1st iteration

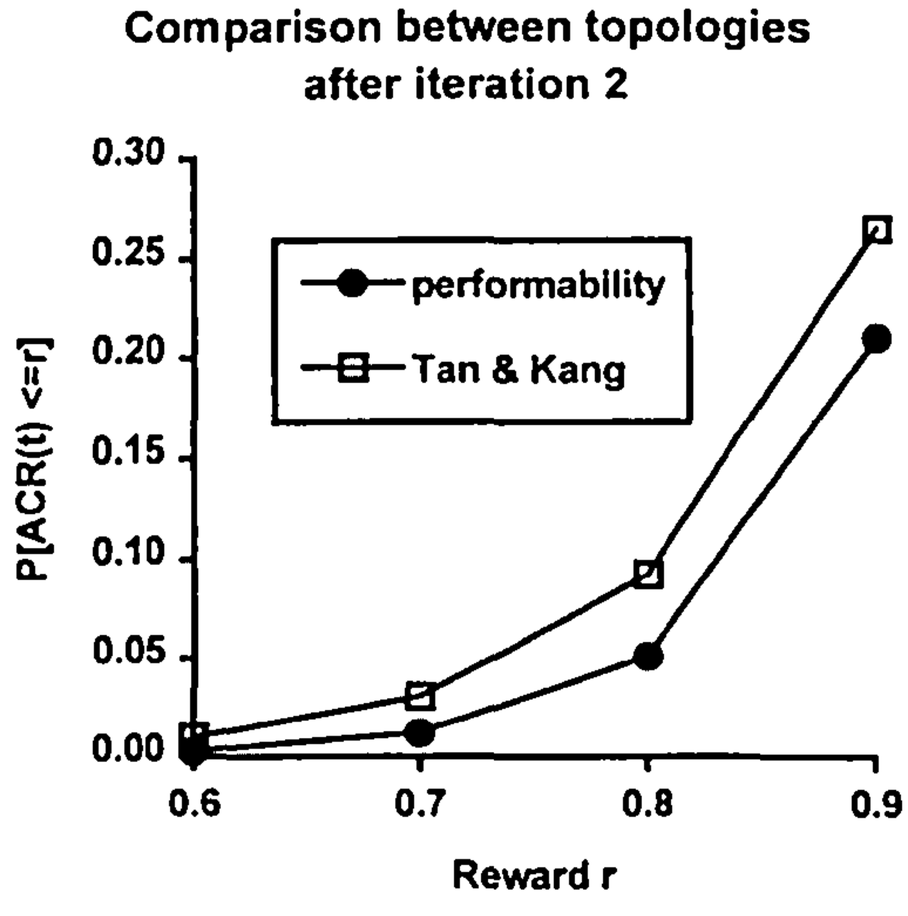


Figure 5.16(b) - Comparison between Kang's and the performability-based approach after the 2nd iteration

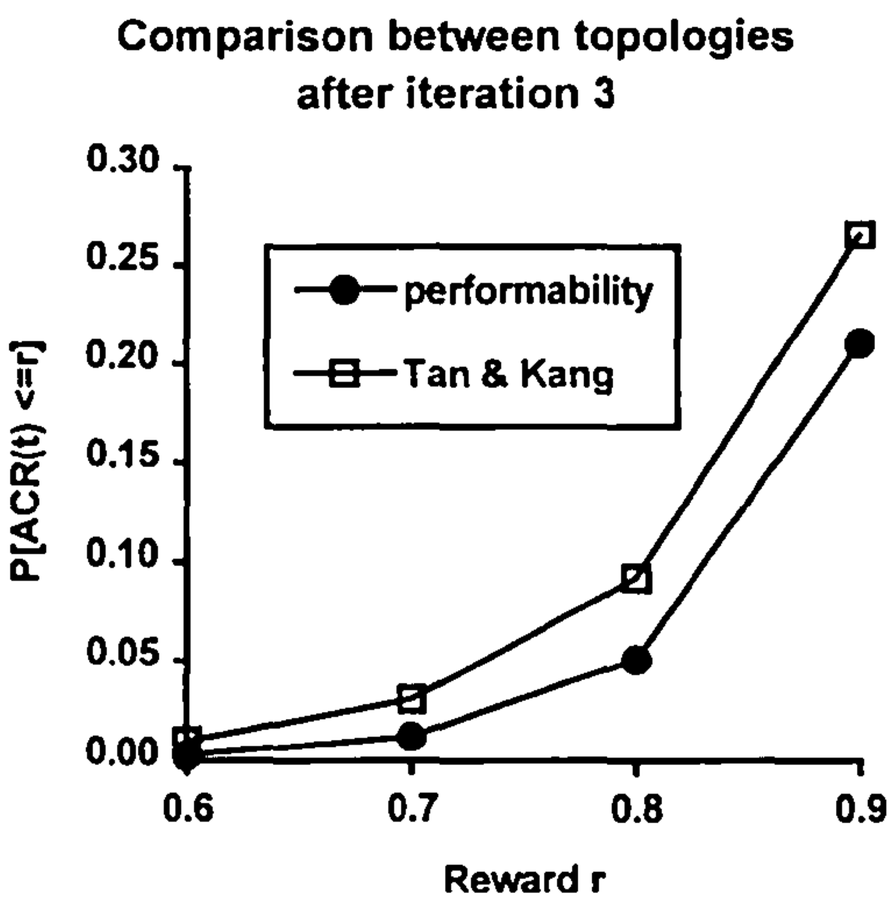


Figure 5.16(c) - Comparison between Kang's and the performability-based approach after the 3rd iteration

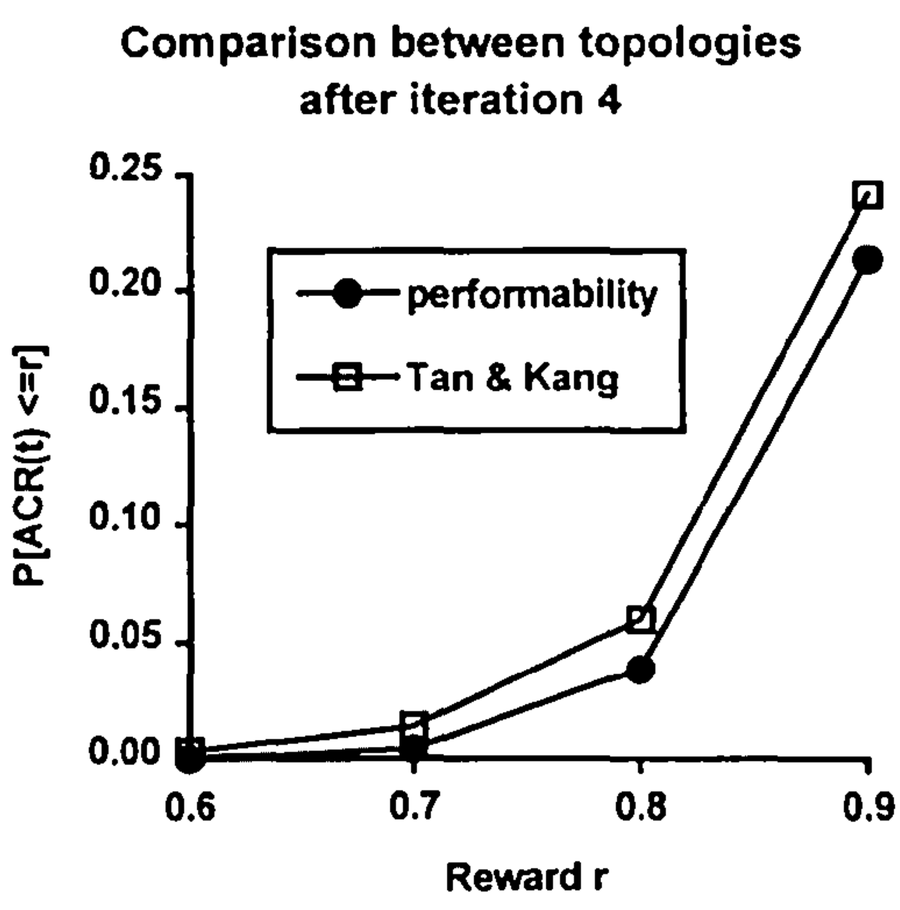


Figure 5.16(d) - Comparison between Kang's and the performability-based approach after the 4th iteration

In all four figures, the network designed using a performability-based approach outperforms the topology generated when Tan and Kang's approach is applied.

At iteration 18, both schemes yield the same network again. The way in which performability varies at the point where reward equals 0.8 as the iterations proceed, starting with iteration 18, is shown in Figure 5.17.

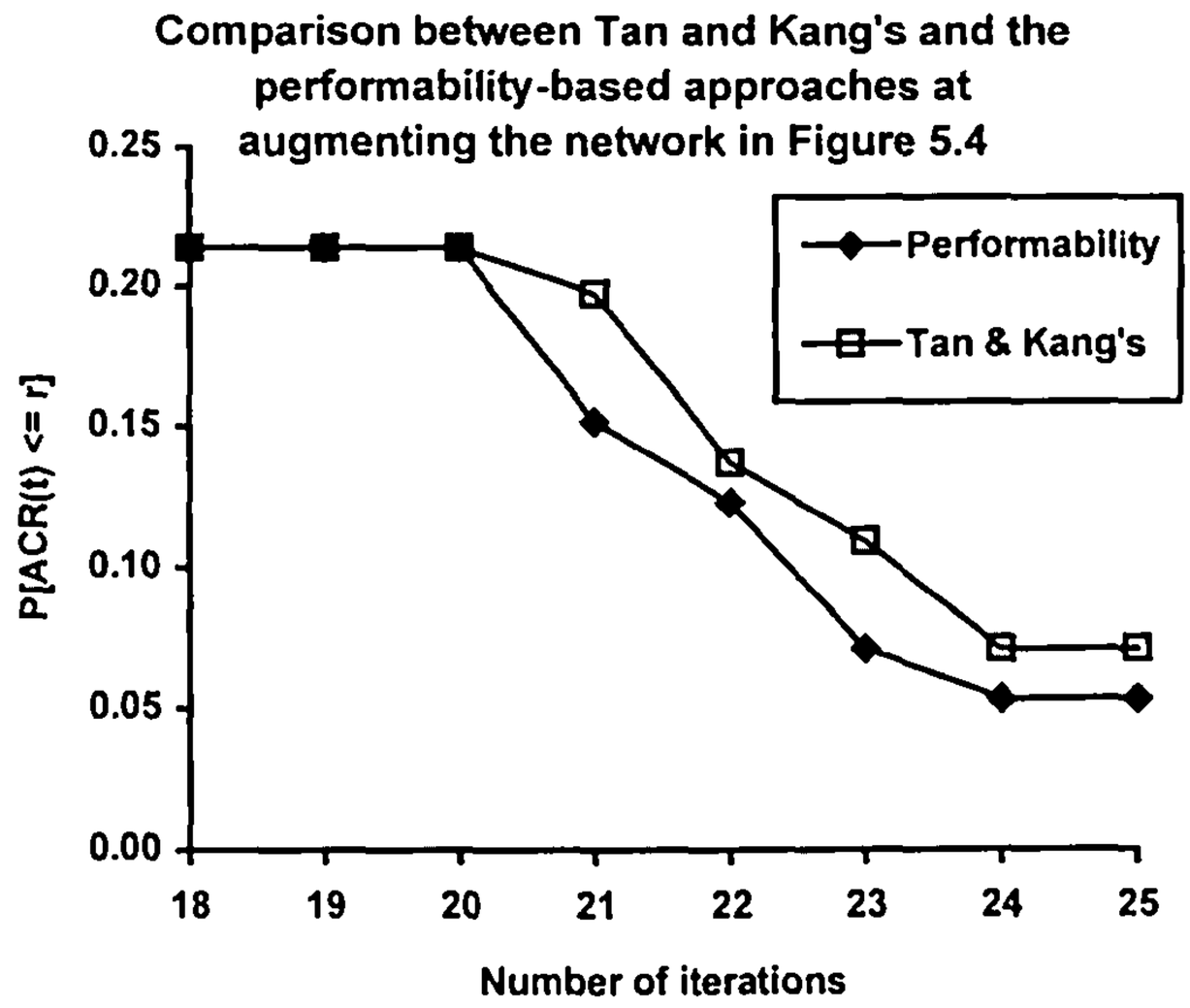


Figure 5.17 - Point performabilities of networks designed using Tan and Kang's and the performability-based approaches per iteration

The networks generated at iterations 19 and 20 are associated with the same reward structure so that they give the same point performability. From iteration 21 onwards, the network designed according to the performability-based approach has a better ability to withstand failures. This is because such a design method aims at selecting the topology that, on average, performs well in all of its failure states, which are weighted by the failure rates and mean repair times associated with them. By choosing the topology performing best in the state where optimisation is applied, Tan and Kang's approach does not ensure that the best improvement is obtained with respect to performance in other failure states as well. It is possible that the topology chosen still performs badly in its other configurations of failure. Note that the degree of improvement when using a performability-based approach depends on the network being designed and the distribution of traffic. In a small network, it is likely that a capacity augmentation has a considerable bearing on the performance in other states. However, in a large network, this is not necessarily true so that a performability-based design, which takes into consideration performance in all states of failure, is expected to produce networks that are even more fault-tolerant, and will do so faster.

5.8 Comparison of the design approaches in terms of performance

So far, design approaches have been compared in terms of performability. In this section, the way in which performance in the fully operational state varies as the iterations proceed is investigated. In Figure 5.18, results are shown for networks designed using increases in connectivity as augmentation mechanism. Capacity augmentation is employed in generation of results in Figure 5.19. The networks being compared correspond to the networks that have already been compared in Sections 5.4 and 5.5 in terms of performability.

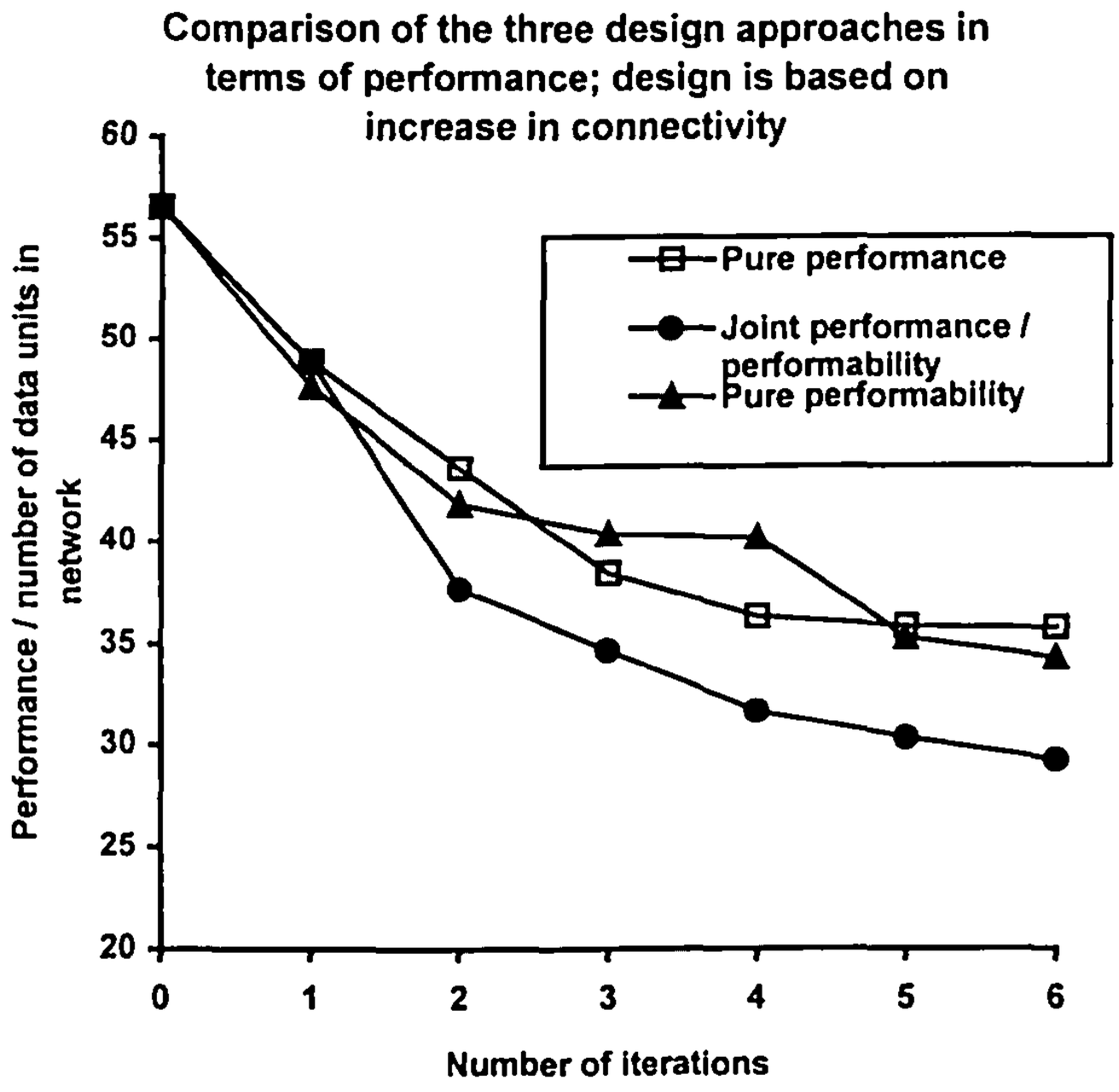


Figure 5.18 - Performance of networks designed using increase in connectivity, in the fully operational state

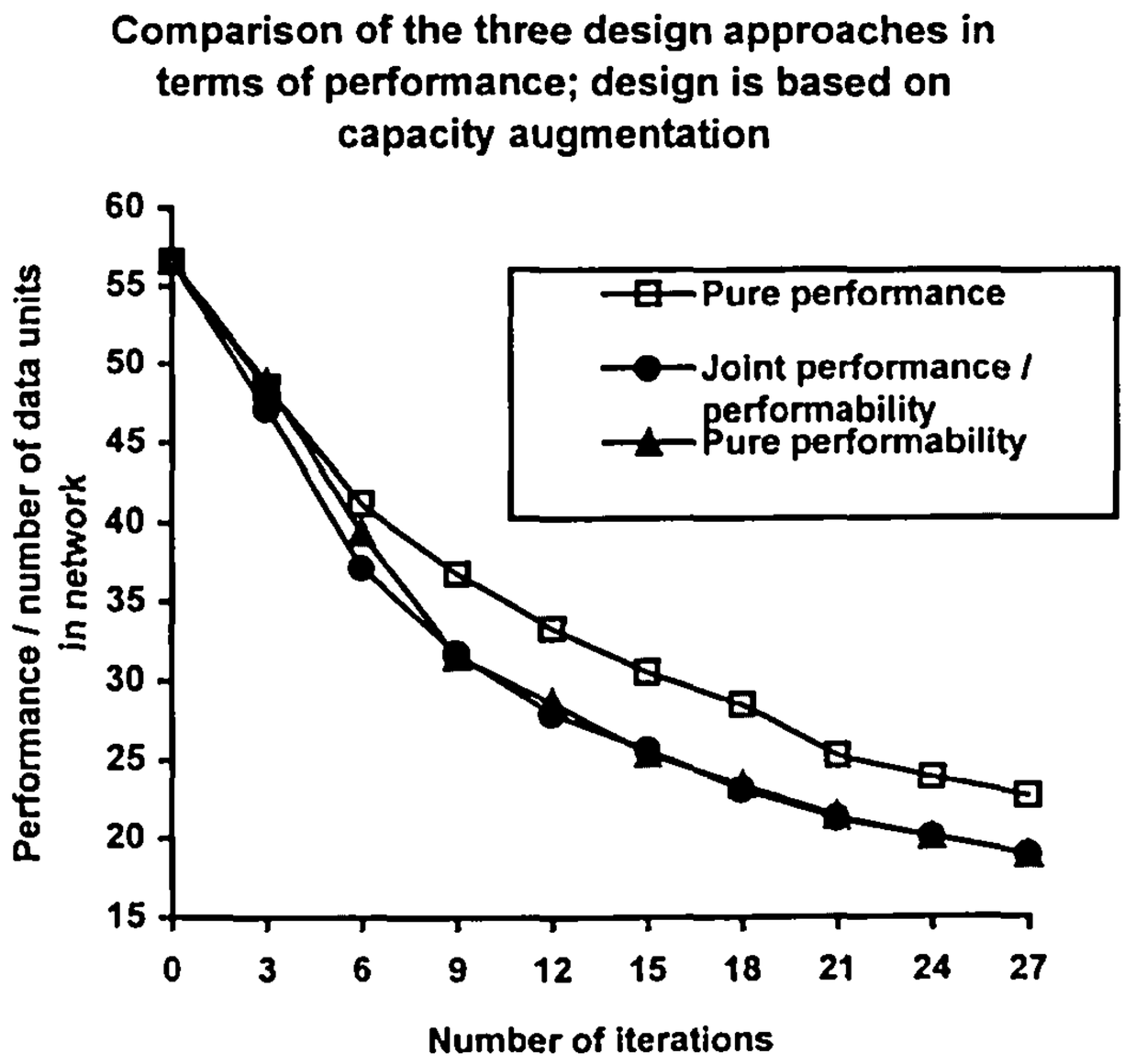


Figure 5.19 - Performance of networks designed using capacity augmentation, in the fully operational state

Interestingly, although the pure performance approach aims at optimising performance in the fully operational state, and the performability-based approaches do not, in this specific example, these schemes, the joint performance/performability and pure performability approaches, generate networks which deliver a better performance when all components are working. In Figure 5.18, the pure performance and pure performability approaches alternate in producing a better network as the iterations proceed. However, in Figure 5.19, the performability-based approaches outperform the conventional design technique at all iterations. The argument is that although performability-based methods optimise performance in states of failure, this does not necessarily mean that they are detrimental to the rate at which performance is improved in the fully operational state.

5.9 Conclusions from comparing the different design approaches

The results have shown how a capacity augmentation and an increase in connectivity result in an improvement in the performance of a network in its various states. Basically, both mechanisms contribute to an increase in the overall capacity of the network, and under

conditions where traffic demands remain the same, and optimal routing is used, performance is expected to improve. Such a statement is applicable to any network when more capacity is put in, provided the extra capacity is utilised.

Given the above fact, the question of designing a fault-tolerant network addresses the issue of where to add redundancy, or extra capacity, to achieve the best network. This amounts to deciding what criteria need to be used in the choice of the state and the best topology. As already explained, traditional design does not take into consideration performance in states of failure, and optimises performance, assuming failure-free conditions. On the other hand, the proposed fault-tolerant design technique not only optimises in a state of failure, but also aims at generating the network which delivers the best average performance over a period of time, when components may fail and need time to be brought back into operation. The examples used have shown that, irrespective of the augmentation mechanism used, performability-based design approaches do generate a better network in terms of its ability to withstand failures.

In order to illustrate how this occurs, consider the following sample path, which is taken to represent the behaviour of a network over the observation period considered, as it changes state as component failures occur. An upward transition corresponds to failure of a component, whereas a downward transition represents repair. Performance is plotted against time. Assume that the measure is such that the lower the value, the better the performance. Performance is plotted as a steady-state value so that any transients are not shown.

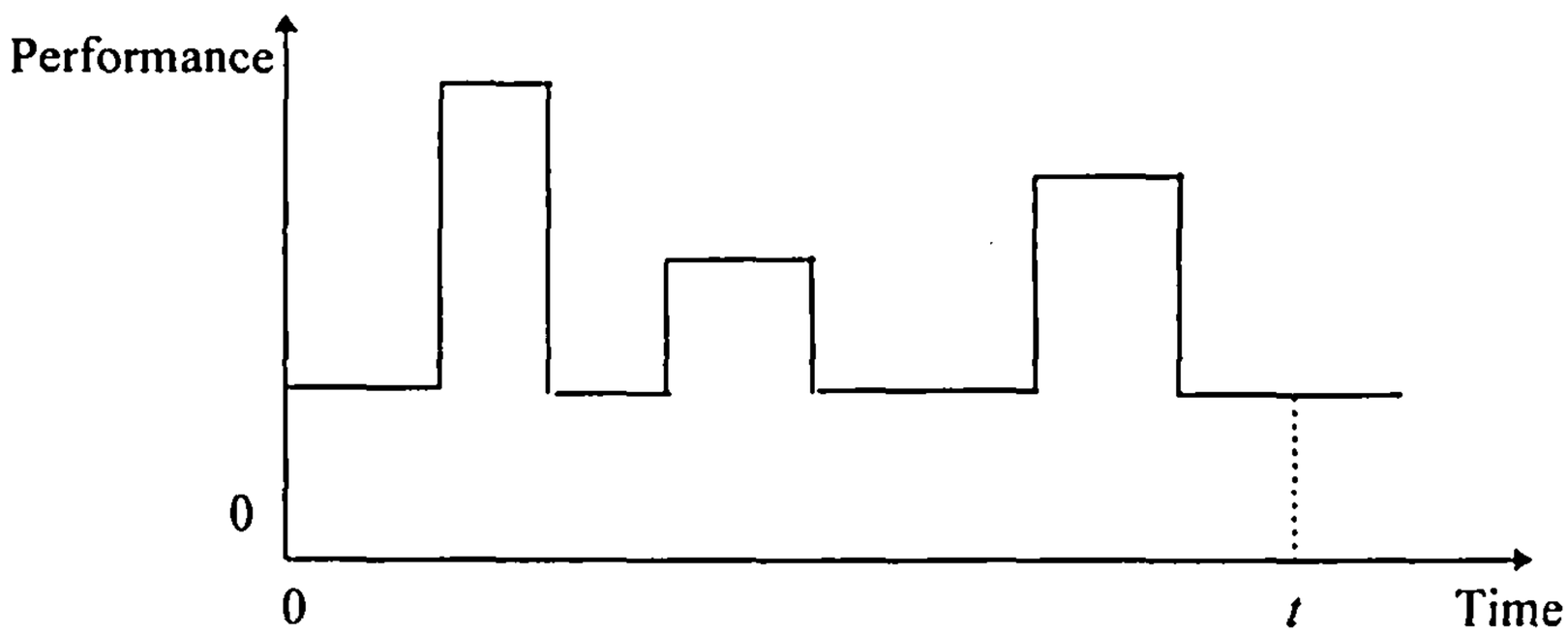
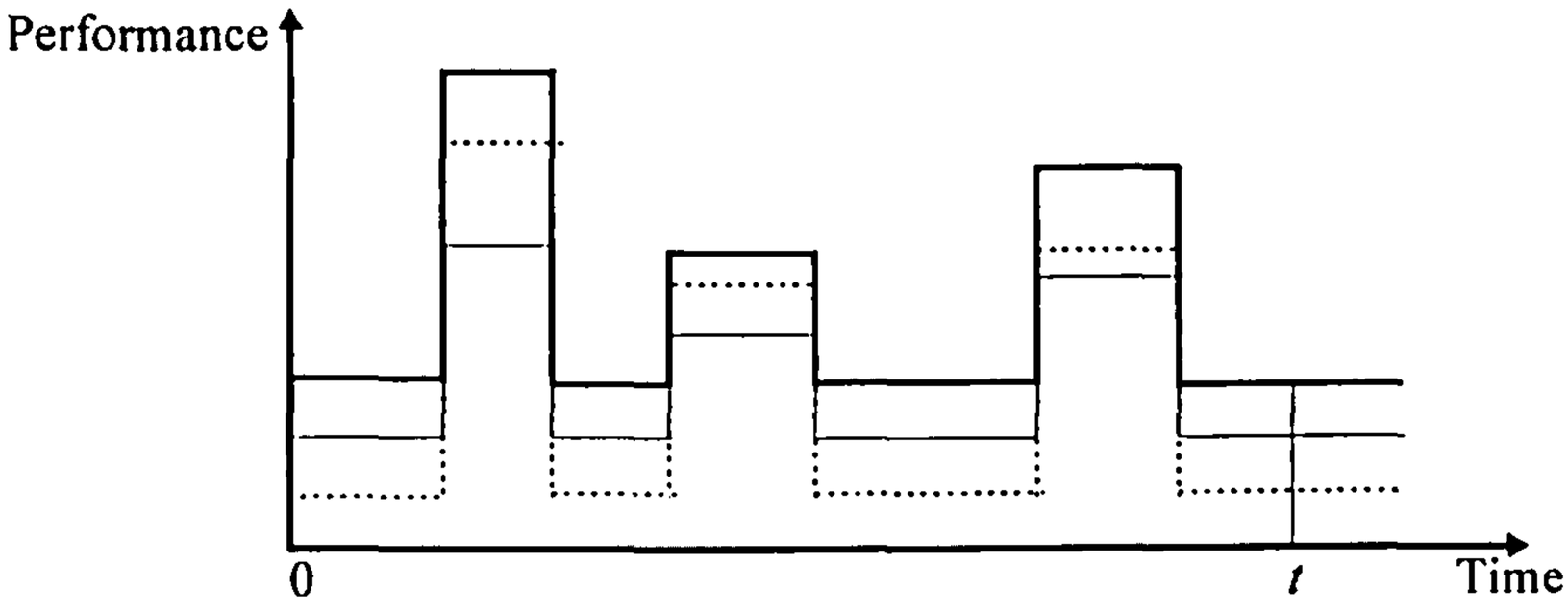


Figure 5.20 - Sample path for a network

Consider the effect of a capacity augmentation, and assume that the path indicated by the dotted line in Figure 5.21 is obtained when the network is augmented according to a *pure performance* approach. Assume that the thin solid line shows the sample path for a performability-based design network. The difference between the two networks is that the one selected according to a performability-based criterion achieves a better performance in its states of failure even if performance in the fully operational state is not the best that can be achieved. Provided the performance when all components are working is acceptable, such a network delivers a better average performance over a period of time.



——— : Sample path for initial network

..... : Sample path for network augmented based on a performance criterion only

——— : Sample path for network augmented based on performability

Figure 5.21 - Sample paths of network after a capacity augmentation

An increase in connectivity contributes to improving performance in a similar way and choice of the topology associated with the best average performance over a period of time ensures that at the end of each iteration, the most fault-tolerant network is obtained.

Such arguments are not affected by the size of the network being considered. In larger networks, optimisation in one particular state may have less bearing on performance in other states. In the worst case, the effect of a capacity augmentation, or addition of a link, is restricted to the state where optimisation is being applied. However, performance cannot be degraded in any other state. The crucial point is that, whatever the size of the network, a

performability-based approach to design selects the topology that performs best when considering the network over a period of time. In so doing, the failure rate, and mean repair time, of components are also being taken into consideration.

In the example considered in Section 5.5, it was shown that, using a performability-based approach to design, a more fault-tolerant network could be obtained on the addition of fewer links than when a pure performance approach is taken. This result is due to the fact that the resources are distributed in a better way across the network so that performance in the states of failure is better even though the network's overall capacity is less. Although this result cannot be generalised in the sense that it cannot be said that a more fault-tolerant network is always designed with fewer resources, the example adds weight to the viability of the performability-based design approach.

The benefits of using performability in the choice of the topology are very clear, but when it comes to choice of the state, using a performability index does not always yield better results. It was expected that by indicating which is the most important component with respect to its rate of failing, mean repair time required, and degree of degradation caused upon failing, a performability-based choice of state would contribute to designing a fault-tolerant network more than a performance-based choice. However, the reason this does not always happen is because any augmentation in a particular state has an effect on performance in all the states in which the network can be. The way in which a particular augmentation mechanism affects performance in other states does not depend on the criterion used to select the state where optimisation is investigated.

On the whole, by virtue of the way in which the best topology is chosen, a performability-based design technique does generate more fault-tolerant networks, and in some cases, may even do so using reduced resources. Although it does not optimise performance in the fully operational state, the proposed design approach does not necessarily adversely affect the rate at which performance is improved in that state. The price paid is, however, the amount of computation involved in a performability-based design. In conventional network design, only one state is considered, where no failures are allowed. Therefore, if n possible ways of augmenting the network need to be compared,

performance has to be evaluated in n states, which are only the fully operational state for each resulting configuration. If design is based on performability, and if k failure states are allowed, then for each possible configuration, performance has to be evaluated for $(k+1)$ states, giving a total of $n(k+1)$ states. Furthermore, comparison between the networks requires evaluation of their performability, where computation depends on the size of the network, length of the observation period, number of different rewards considered, and maximum rate Λ chosen, as explained in Chapter 3. Therefore, as the size of the network to be designed increases, the benefits of using the performability-based approach are offset by the computational and memory requirements. However, techniques of state space reduction can be applied to the performability model in order to deal with this problem.

At the expense of more time and computation going into design and planning, a network which can deliver a good performance level when all its components are working, and an acceptable level under conditions of failure can be designed. This network is definitely better than one designed more quickly and easily, but where a single component failure may have significant effects on performance.

5.10 Comparison between adding a link and augmenting capacity of an existing link

It is stated in [BER87] that when the cost function is based on the M/M/1 approximation, typically *for low input traffic, each source-destination pair tends to use only one path for traffic, the fastest path in terms of packet transmission times, and as traffic input increases, additional paths are used to avoid overloading the fastest path.* Such a statement suggests that capacity augmentation would be favoured, especially under conditions of low traffic. The question addressed in this section is whether any of the augmentation mechanisms is better.

Given a network, the aim of the routing algorithm is to distribute the flows in such a fashion as to minimise the cost function defined as $\sum_{k\ links} D_k(F_k)$ for all k links constituting the network where

$$D_k(F_k) = \frac{F_k}{C_k - F_k} \text{ and}$$

C_k is the transmission capacity of link k and

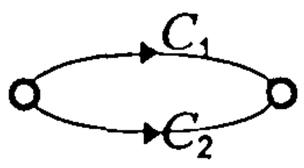
F_k is the traffic flow on link k.

Therefore, if a network consists of n links, the cost function to be minimised is as follows:

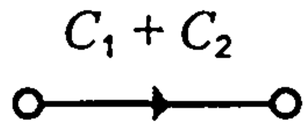
$$\sum_{k\ links} D_k(F_k) = D_1(F_1) + D_2(F_2) + \ \ldots\ D_n(F_n).$$

Upon a capacity augmentation, the flows are redistributed among the existing links. The cost function is still comprised of n terms. When a link is added, an extra term is included in the summation to give $(n + 1)$ terms. This additional term may favour a capacity augmentation to addition of a link.

A 2-node network is first used to illustrate this. Two networks are to be compared. In configuration *A*, the nodes are connected by two links of capacities C_1 and C_2, giving a total capacity of $C_1 + C_2$ and in configuration ***B***, there is a single link of capacity $(C_1 + C_2)$. In terms of the capacity between the two nodes, configurations ***A*** and ***B*** are equivalent.



Configuration ***A***



Configuration ***B***

Let total flow of traffic be equal to F. In configuration *A*, flow is distributed with f_1 units going through the link with capacity C_1 and f_2 units going through the link with capacity C_2. The total flow is carried by the single link in configuration ***B***. The cost functions for each network are as follows:

Network ***A*** : $D_A = \frac{f_1}{C_1 - f_1} + \frac{f_2}{C_2 - f_2}$ (5.1)

Network ***B*** : $$D_B = \frac{(f_1 + f_2)}{(C_1 + C_2) - (f_1 + f_2)} \tag{5.2}$$

Subtracting D_B from D_A and re-arranging terms gives

$$D_A - D_B = \frac{f_1(C_2 - f_2)}{(C_1 - f_1)\left[(C_1 + C_2) - (f_1 + f_2)\right]} + \frac{f_2(C_1 - f_1)}{(C_2 - f_2)\left[(C_1 + C_2) - (f_1 + f_2)\right]} \tag{5.3}$$

Given that $C_1 > f_1$ and $C_2 > f_2$,

$$D_A - D_B > 0.$$

This implies that, assuming that all links are operating, configuration ***B*** is always associated with a better performance. However, failure of the link causes drastic degradation in performance because no communication is possible at all. In configuration ***A***, in case of a single link failure, communication is still possible via one link and performance does not degrade that badly. In terms of performability, configuration ***A*** is better, assuming that failure and repair rates of all links are the same.

To generalise to a larger network, consider a network of n links, so that the cost function to be minimised is as shown in equation (5.4).

$$\sum_{k=1}^{n} \frac{F_k}{C_k - F_k} = \frac{F_1}{C_1 - F_1} + \frac{F_2}{C_2 - F_2} + \ldots + \frac{F_n}{C_n - F_n} \tag{5.4}$$

Let the flow F_n on link n be denoted by $\left(F_n^{\bullet} + \Delta F_n\right)$. Assume that this link is the most loaded. The possible augmentation mechanisms consist of increasing the capacity of link n, or adding another link to deviate traffic from it. In order to simplify the analysis, assume that the extra link is added between the end nodes of link n so that the sum of terms for the other (n - 1) links does not change.

The term of interest is the one corresponding to link n. Based on this assumption, the following cost functions can be derived.

When the capacity of link n is increased by ΔC_n, the resulting cost function is

$$D_1 = \sum_{k=1}^{n} D_k(F_k) = \sum_{k=1}^{n-1} D_k(F_k) + \left(\frac{F_n^* + \Delta F_n}{(C_n + \Delta C_n) - (F_n^* + \Delta F_n)} \right). \tag{5.5}$$

Assume that when a link of capacity ΔC_n is added to the network, ΔF_n units of traffic are shifted to this link so that the cost function is given by

$$D_2 = \sum_{k=1}^{n} D_k(F_k) = \sum_{k=1}^{n-1} D_k(F_k) + \left(\frac{F_n^*}{C_n - F_n^*} \right) + \left(\frac{\Delta F_n}{\Delta C_n - \Delta F_n} \right). \tag{5.6}$$

Subtracting D_2 from D_1 and re-arranging terms,

$$D_2 - D_1 = \frac{F_n^* (\Delta C_n - \Delta F_n)}{(C_n - F_n^*)((C_n + \Delta C_n) - (F_n^* + \Delta F_n))} + \frac{\Delta F_n (C_n - F_n^*)}{(\Delta C_n - \Delta F_n)((C_n + \Delta C_n) - (F_n^* + \Delta F_n))} \tag{5.7}$$

Given that $C_n > F_n^*$ and $\Delta C_n > \Delta F_n$,

$D_2 - D_1 > 0$.

The positive difference shows that performance is better when the capacity of an existing link is augmented rather than when an extra link is added. However, in this discussion, it has been assumed that the augmentation mechanism does not affect the sum of terms corresponding to the other (n - 1) links in the network, which is not necessarily true. Any augmentation applied to the network can affect flows of traffic on all the links in order to achieve the best performance. This simple analysis shows that a capacity augmentation would tend to be favoured to addition of a link; but, in practice, due to redistribution of flows over all links, such a statement is not always valid. With larger networks, the optimal routing problem becomes more involved to solve so that it is very difficult to predict the performance analytically. A computational solution is required.

Furthermore, as shown in the 2-node example, even if a capacity augmentation is preferred to the addition of a link to improve performance in one state, it might not be the best mechanism when failure scenarios are taken into account. Assuming that load needs to be eased for link *n*, adding an extra link to the network not only ensures that performance is enhanced in the state where the network is being optimised, but it also contributes to providing an additional path for traffic when link *n* fails. However, if the capacity of link *n* is increased, there is no improvement in performance when that link fails.

Therefore, firstly, for large networks, it is not possible to decide which mechanism would provide the best improvement in performance in the state being optimised. Secondly, even if one mechanism gives the best performance in one state, it does not necessarily mean that it is the better mechanism with respect to performance in all states of failure. Both augmentation techniques need to be used in the *fault-tolerance augmentation* stage.

5.11 Set E - Comparison between increase in connectivity and capacity augmentation

The relative merits of the two augmentation mechanisms are assessed by comparing the networks designed when each of the technique is used separately. The three approaches to design are investigated, namely the pure performance approach, the pure performability approach and the joint performance/performability approach. The initial network, which is to be subjected to augmentation, is the topology given in Figure 5.4.

5.11.1 Pure performance-based design.

In this conventional design scheme, augmentation is applied in the fully operational state in order to achieve the best performance in that state. The way in which the two augmentation mechanisms affect performance in that state per iteration is shown in Figure 5.22. Note that addition of a link effectively means that two unidirectional links are established between the connecting pair of nodes, whereas capacity augmentation is applied to one unidirectional link at a time. Therefore, the network obtained after each iteration of an increase in connectivity-based design, is compared with the network generated after every two

iterations when capacity augmentation is used. This ensures that the networks being compared have equal overall capacity.

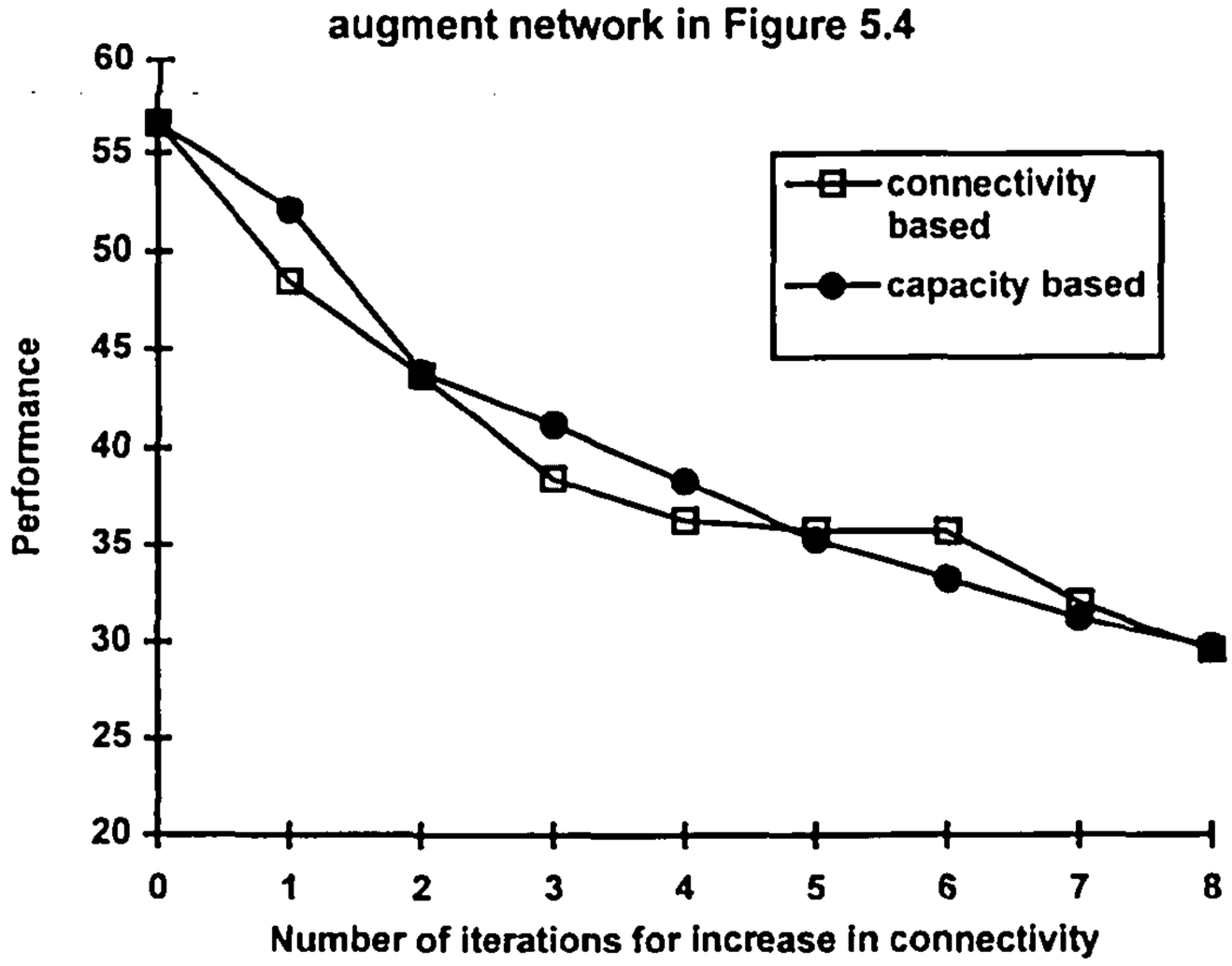


Figure 5.22 - Performance of networks designed using each mechanism according to a pure performance approach

In Figure 5.23, the distribution of average accumulated reward for the networks generated when augmentation employs each of the two mechanisms is shown. The networks being compared correspond to the topology designed after 8 iterations of an increase in connectivity-based method, and the network obtained after 16 capacity augmentations.

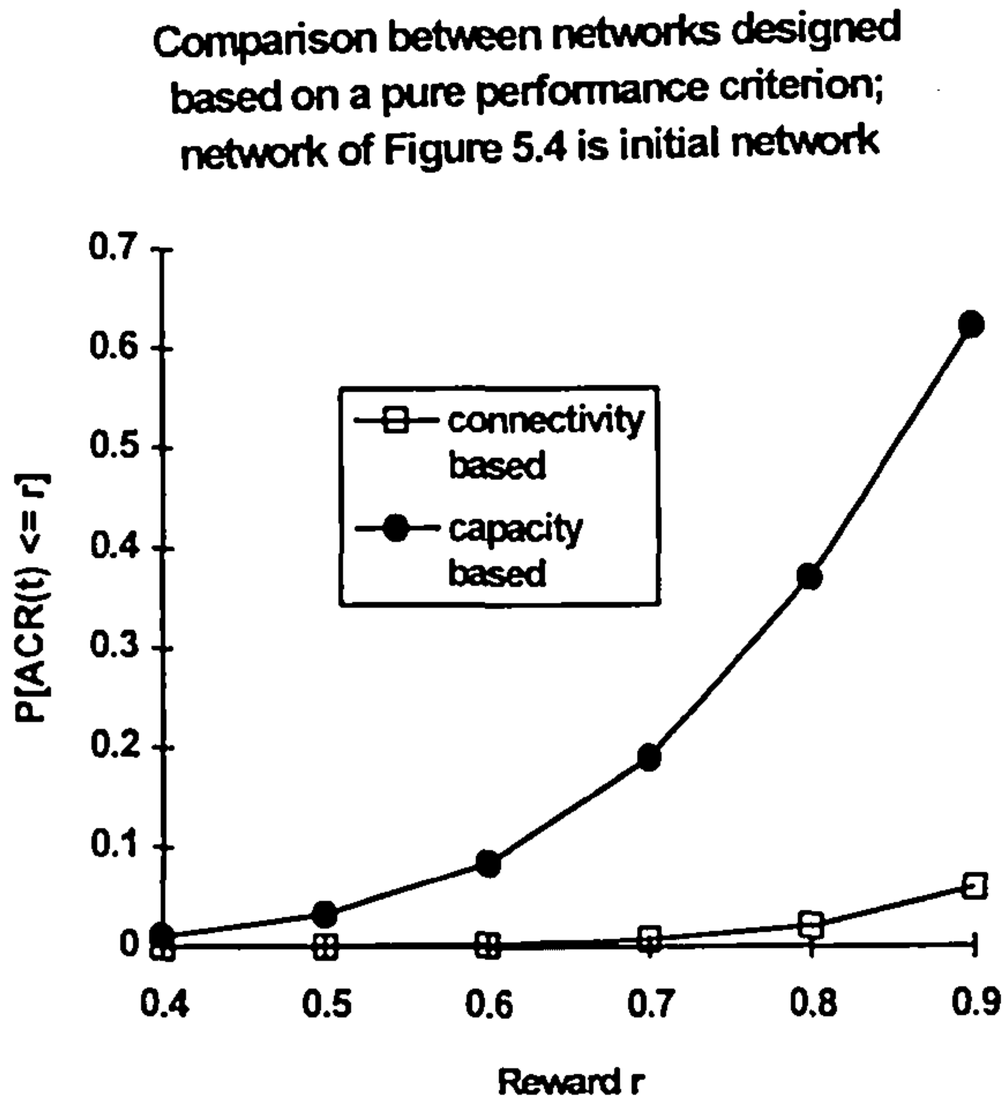


Figure 5.23 - Performability distributions of networks designed using each mechanism according to the pure performance approach

According to results shown in Figure 5.22, generally, the addition of extra links seems to be more beneficial in improving performance in the fully operational state. Capacity augmentation does give better results between iterations 5 and 7. In terms of performability, increase in connectivity generates a better network as shown in Figure 5.23.

5.11.2 Joint performance / performability - based design

Comparison is done between the network designed after each iteration of an increase in connectivity-based approach and the network generated after two iterations of a capacity augmentation-based approach. Results are obtained in terms of performability at the point where reward equals 0.8, that is P[$ACR(t) \leq 0.8$], for t taken to be 10 days, and are shown in Figure 5.24.

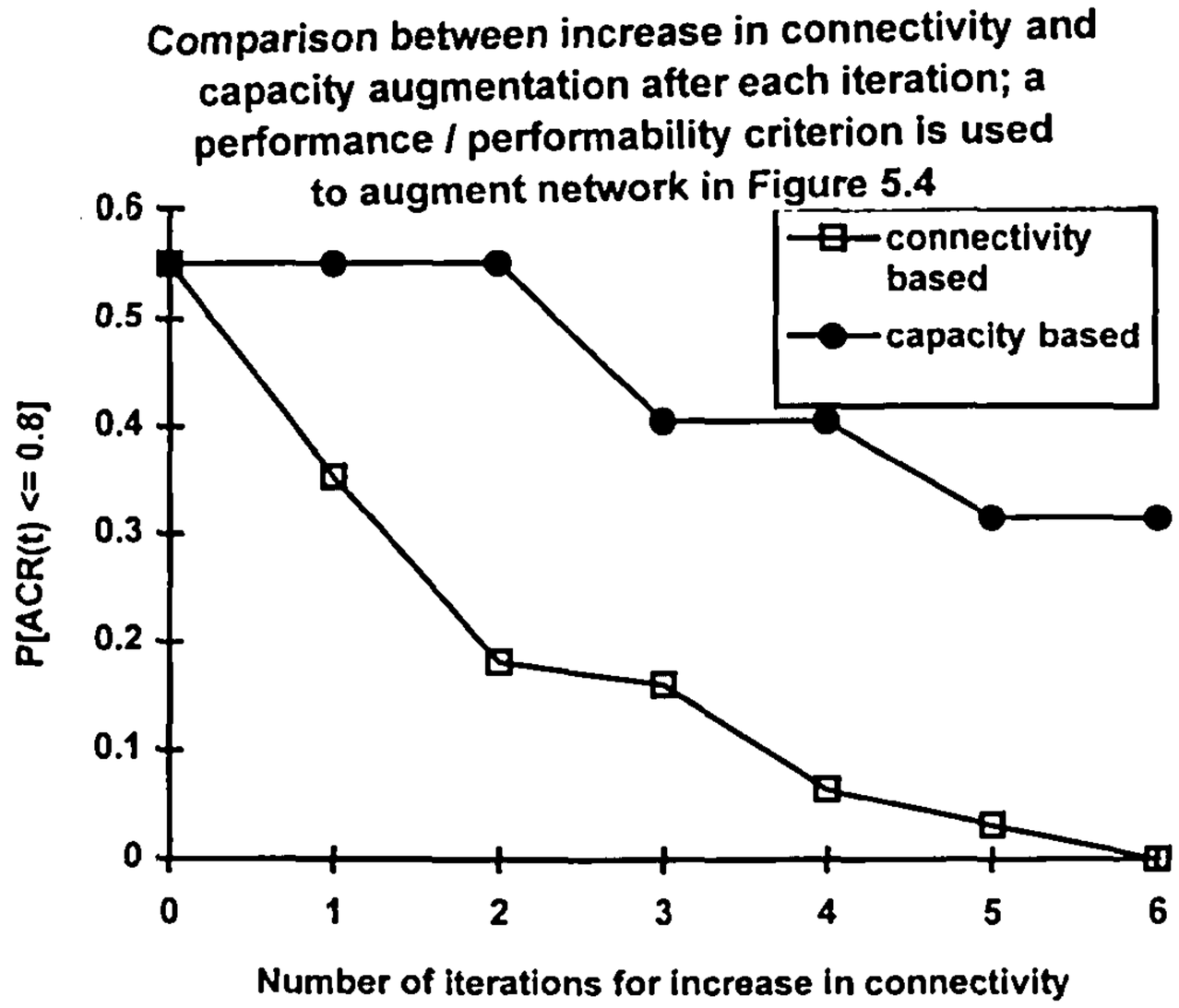


Figure 5.24 - Point-performabilities of networks designed using each mechanism according to the joint performance / performability approach

The final networks obtained when each augmentation mechanism is applied, are now compared. The stopping criterion in each design is the point when all performance levels are less than 150, so that based on the first definition of reward, the performability distribution of each network is the ideal step function. Comparison is carried out, with rewards derived according to the second definition, giving the distributions shown in Figure 5.25.

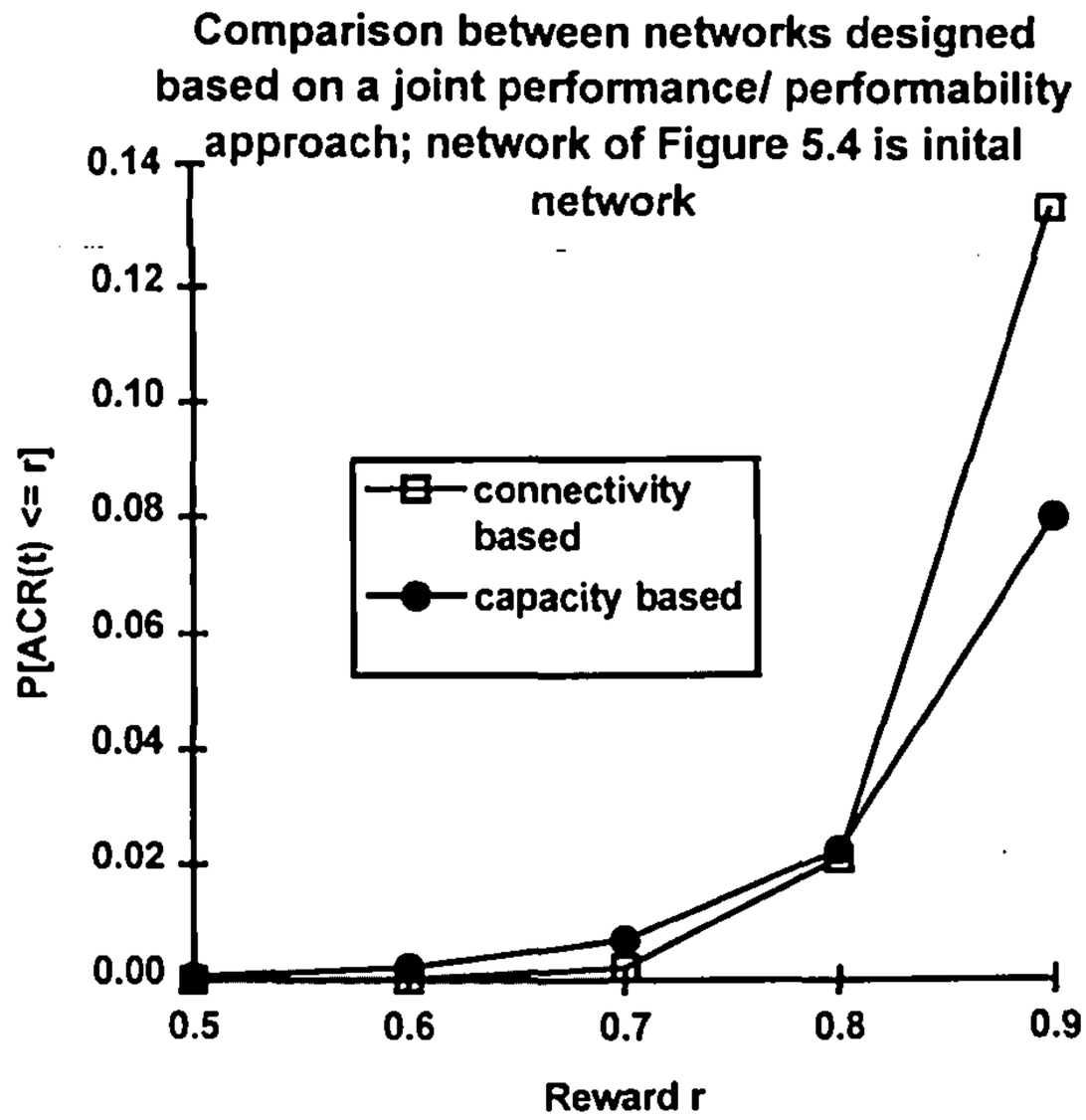


Figure 5.25 - Performability distributions of networks designed using each mechanism according to a joint performance / performability approach

5.11.3 Pure performability-based design

A similar analysis is carried out with networks designed according to a performability-based choice of state and topology method. Results are as shown in Figures 5.26 and 5.27. In Figure 5.26, performability taken at the point where reward equals 0.8 is plotted against the number of iterations. The distributions of average accumulated reward for the final networks generated using each augmentation mechanism are shown in Figure 5.27.

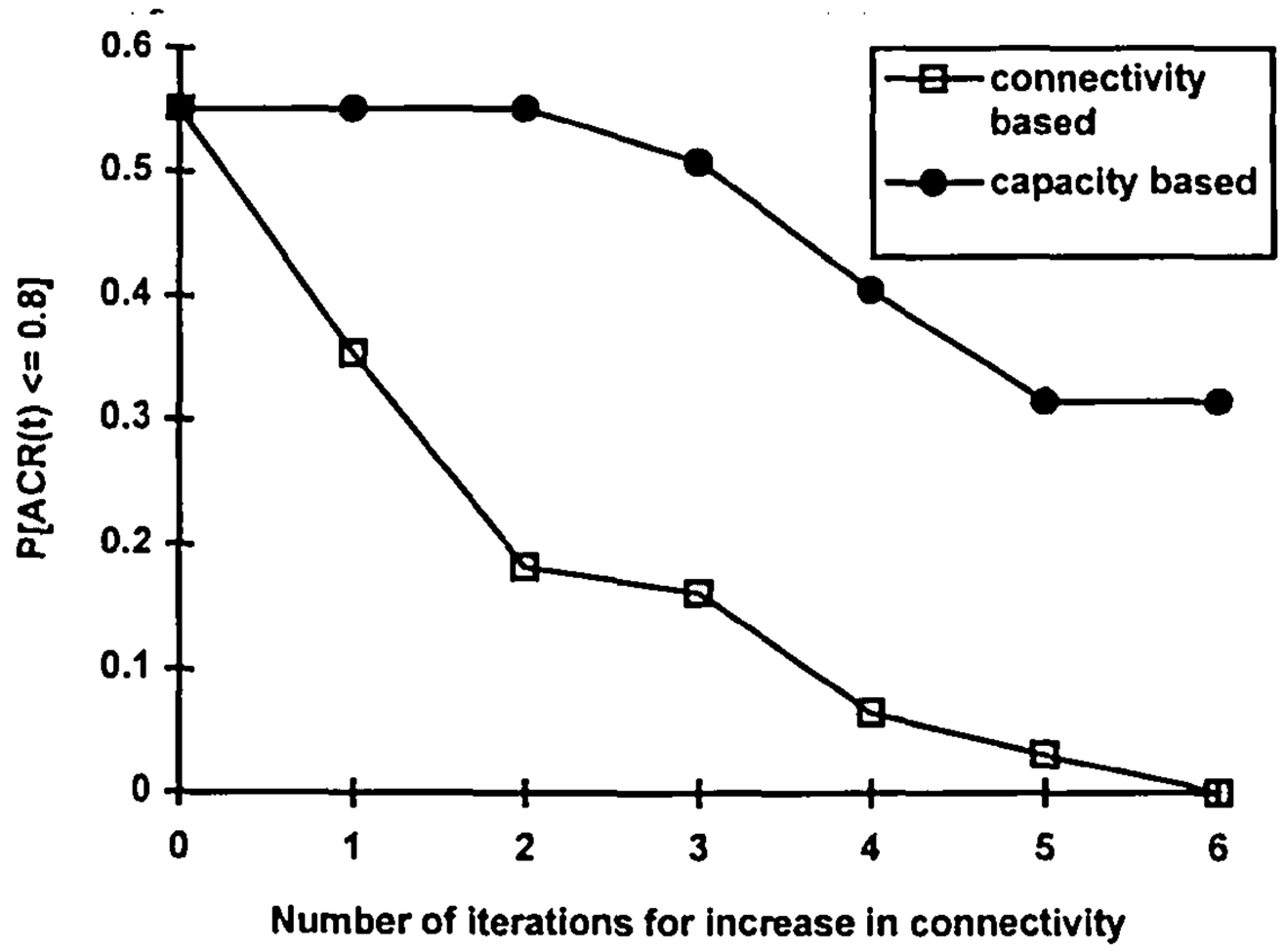


Figure 5.26 - Point-performabilities of networks designed using each mechanism according to the pure performability approach

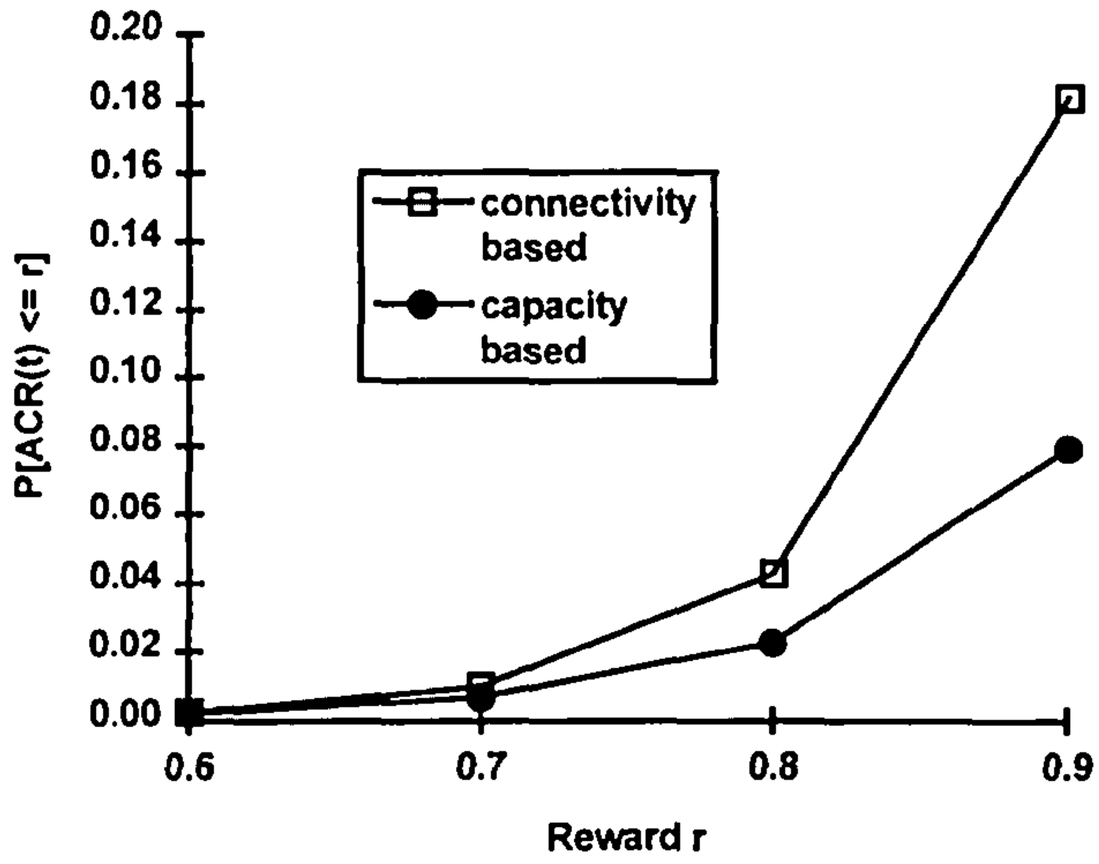


Figure 5.27 - Performability distributions of networks designed using each mechanism according to the pure performability approach

Based on the results presented in this section, there is no reason for stating that either of the augmentation mechanisms produces a better network. In the pure performance-based design, capacity augmentation generates a more fault-tolerant network, as shown in Figure 5.23. However, when comparing performance in the fully operational state, the relative benefits of using each mechanism are not clear.

In the performability-based approaches, increase in connectivity is preferable to capacity augmentation in the sense that the networks generated are associated with a performability, which converges faster to the ideal value. A comparison of final networks designed shows that a capacity augmentation-based technique produces a more fault-tolerant network, as can be seen in Figure 5.27. This is less apparent in Figure 5.26. When design is based on capacity augmentation, more iterations are required for performability specifications to be met. As a result, the final network is equipped with a higher overall capacity than one augmented through the addition of extra links. This explains why, in this example, when comparing final networks, capacity augmentation appears to be a better technique. Nevertheless, increasing the connectivity of the network improves performability faster.

A mathematical comparison of the two schemes has shown that under certain assumptions regarding redistribution of traffic, the cost function being optimised in routing may favour capacity augmentation to addition of links in a specific state. Such a statement is not necessarily always valid. However, comparison of the networks designed shows that increase in connectivity tends to be better, from a performability point of view; it produces fault-tolerant networks faster. Furthermore, it is expected that the mechanism which would prove to be more suitable is network specific, and depends on the traffic demands as well. Therefore, both mechanisms have to be combined and tried to achieve the best design most effectively.

5.12 Design using both mechanisms

The way in which a combination of both mechanisms can be used with respect to different sets of traffic requirements is investigated. Three sets of traffic demands are considered,

giving three networks. The same initial network, shown in Figure 5.4, is used in the three designs, in the sense that the node locations are the same, so that the structure of the networks generated after the *increase in connectivity* stage are identical. However, the capacities of the links vary according to the traffic requirements. Performability specifications are set to be the same for the three designs.

In design of network A, the traffic demands are the same as the values used in derivation of results in Set E. Traffic requirements used in design of network B are lower, whereas the demands used for design of network C are the highest. Table 5.3 shows the order in which the augmentation mechanisms are applied for each network with its corresponding traffic demands, during execution of the *fault-tolerance augmentation* stage. The augmentation rules outlined in Section 4.2.5.4 are used to identify the traffic scenario being dealt with and decide on the possible augmentation mechanisms, (that is add links or augment the capacity of links) at each iteration of design. A performance-based choice of state and performability-based choice of topology approach was adopted in the design.

Link (*i-j*) refers to a link connecting nodes *i* and *j*, with flow of traffic being in the direction from node *i* to node *j*. Links (*i-j*/*j-i*) correspond to links between nodes *i* and *j* with bi-directional communication. Although capacity augmentation is applied to one unidirectional link at a time, if augmentation of the capacity of link (*i-j*) is followed by augmentation of the capacity of link (*j-i*) at the next iteration, the two iterations have been combined as *augment links* (*i-j*/*j-i*) in the table.

Augmentation mechanisms for Network A	**Augmentation mechanisms for Network B**	**Augmentation mechanisms for Network C**
Add links (2-5/5-2)	Add links (5-10/10-5)	Augment links (2-6/6-2)
Add links (2-7/7-2)	Add links (2-5/5-2)	Add links (4-10/10-4)
Add links (1-10/10-1)	Augment links(5-10/10-5)	Add links (2-10/10-2)
Add links (4-7/7-4)	Augment link (2-1)	Augment links (2-6/6-2)
Augment link (1-10)	Add links (2-4/4-2)	Augment links (6-7/7-6)
Augment links(7-10/10-7)	Augment link (7-6)	Augment links(4-10/10-4)
	Augment link (6-5)	Augment links(2-10/10-2)
		Augment links(2-10/10-2)
		Augment links(4-10/10-4)
		Augment links (2-6/6-2)
		Add links (1-7/7-1)
		Augment link (6-5)
		Augment link (4-10)
		Augment link (7-1)
		Augment link (10-2)
		Augment link (6-5)
		Augment link (4-10)
		Augment link (10-7)

Table 5.3 - Order of application of augmentation mechanisms to network in Figure 5.4

From the above table, addition of links is the main augmentation mechanism used in design of network A. It is complemented in the last iterations by augmentation of the capacities of existing links. As far as network B is concerned, both mechanisms prove to be equally important; capacity augmentation is used even in the early iterations. It should be noted that both networks are equipped with the same capacity at the end, but with different distributions of capacity over the links. Design of network C is mainly dominated by

capacity augmentation. In most iterations, this mechanism is more beneficial in improving performability.

Based on this set of results, it is clear that none of the *fault-tolerance augmentation* mechanisms can be claimed to be better. The most appropriate mechanism depends on the structure of the network, the traffic demands and the way flows are being distributed in the network. Therefore, at each iteration, it is best to try the different possibilities of adding links and augmenting capacities according to the rules outlined in Section 4.2.5.4, with the aim of generating the most fault-tolerant network.

Furthermore, an investigation of the behaviour of network A when faced with traffic demands of network B shows that the network has no difficulty coping with lower traffic in all of its states, so that performability specifications are still satisfied. On the other hand, the network's performance, even in the fully operational state, degrades completely when network A is tested with traffic demands of network C. It follows that performance in the failure states is even worse and performability requirements are not met at all.

Typically, over a period of time, traffic demands are variable, with flows becoming more intense during peak hours. By designing the network assuming peak hour traffic demands, it is expected to be adequately equipped to cope with reduced flows of traffic. However, the network is expensive in terms of its resources, both in terms of setting up and maintenance. A trade-off between cost and performability is desirable, for varying traffic.

5.13 Set F - Intermediate states of failures

In the examples considered so far, each link has been assumed to be in one of two states: up or down. To exploit the full concept of performability, intermediate states of failure can also be considered, where a link can be operating at a fraction of its capacity. Such a situation may result from occurrence of some software, or hardware, fault. In this section, the effect of including intermediate states of failure for each link on the design is investigated. Each link is assumed to be in one of three states: the up state when it is operating at full capacity, the down state where it is completely down, and an intermediate state where it is operating

at half capacity. Figure 5.28 shows the Markov chain model for a single link; state l_1 corresponds to the intermediate state of failure, state l_0 the up state, and l_2 the down state. Transitions are allowed between all the states, so that a transition from state l_0 to state l_1 indicates partial failure, and a transition from state l_2 to state l_1 corresponds to partial repair of the link.

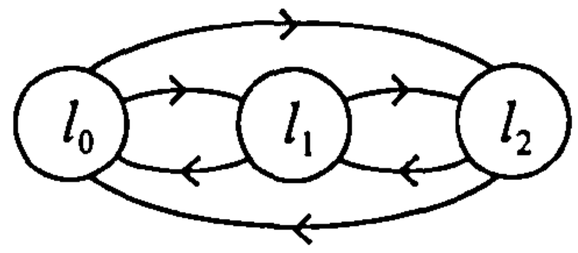


Figure 5.28 - Markov chain model for a single link

Assuming this new failure scenario, the *fault-tolerance augmentation* stage is executed, with the initial network corresponding to Figure 5.4. The different approaches to design, according to the criteria used in choice of state and topology, are investigated. A combination of increase in connectivity and capacity augmentation is used to improve the fault-tolerance of the network. In Figure 5.29, results are presented in terms of the performability distribution of the network generated after 5 iterations of each design. Performability is evaluated over a time period of 10 days. Figures 5.30(a) and (b) show how point performability at reward equal to 0.8 varies as the iterations proceed.

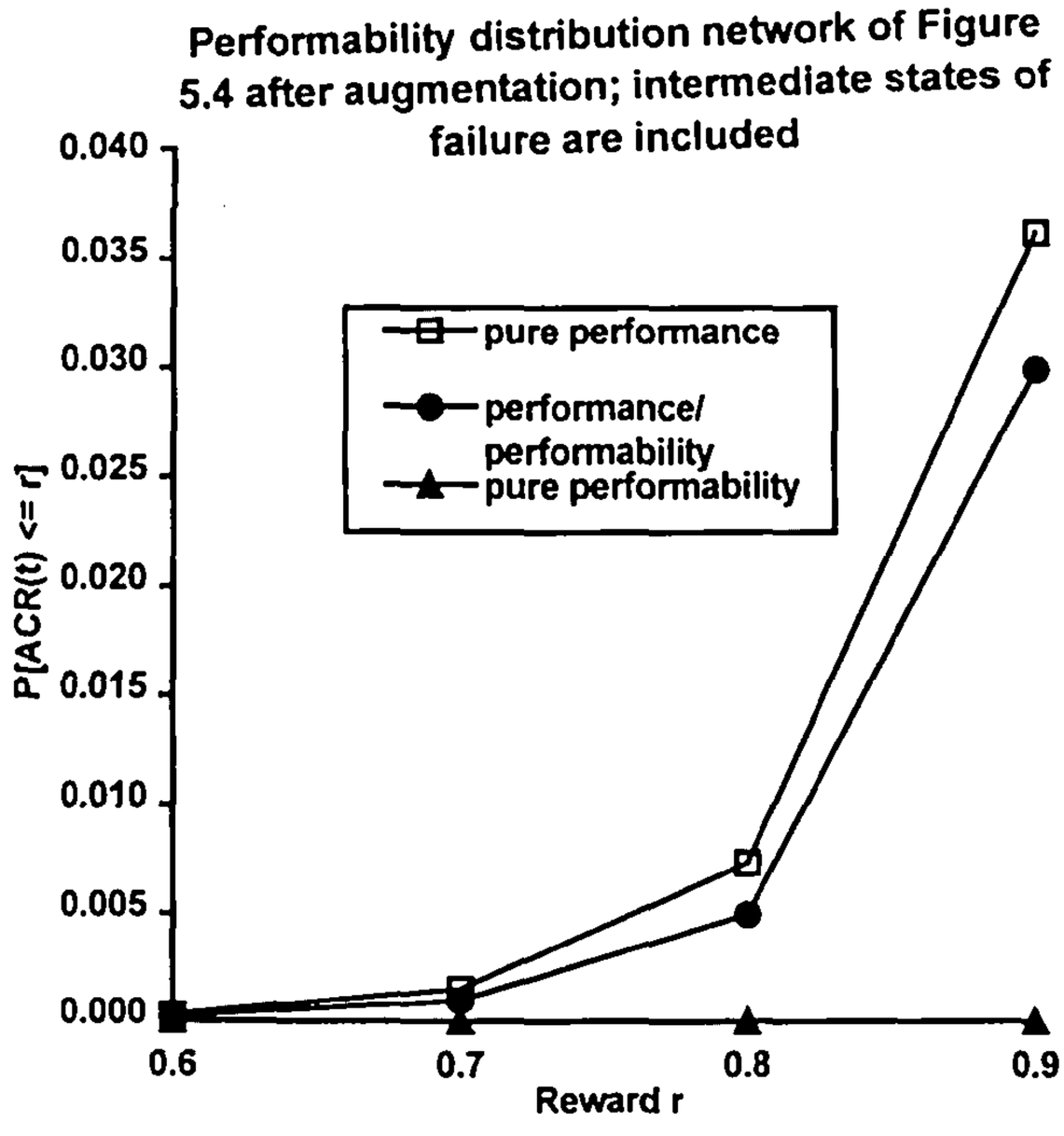


Figure 5.29 - Performability distributions of networks when intermediate states of failure are considered

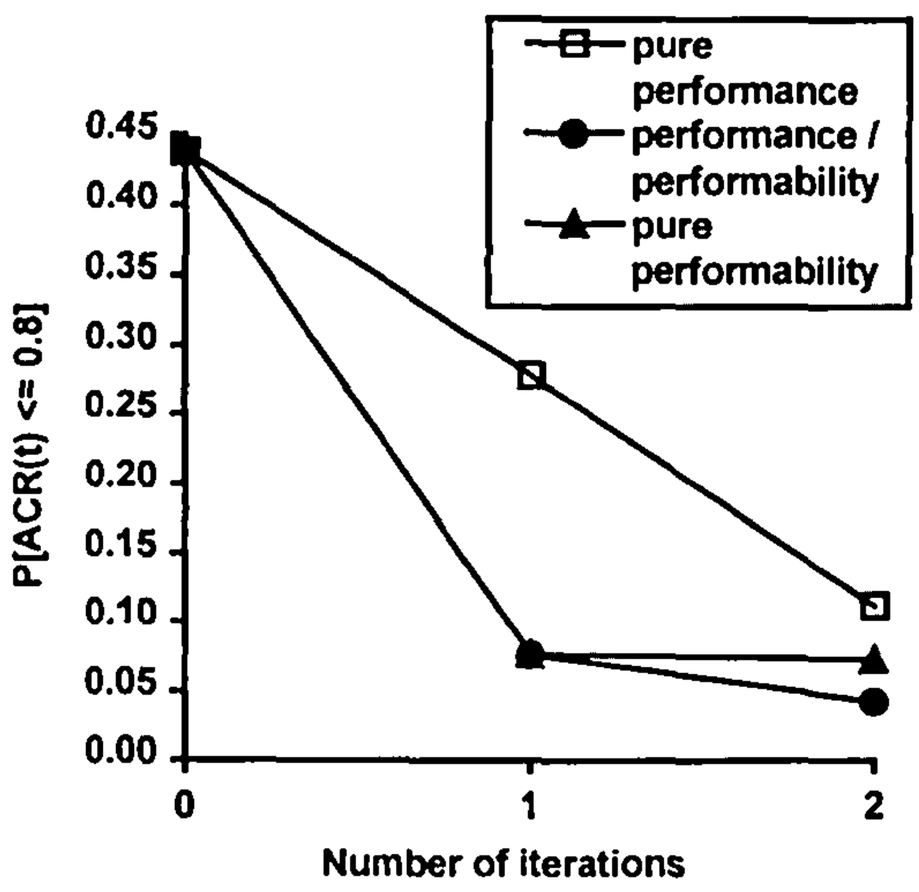


Figure 5.30(a) - Point-performability of networks for iterations 0 to 2

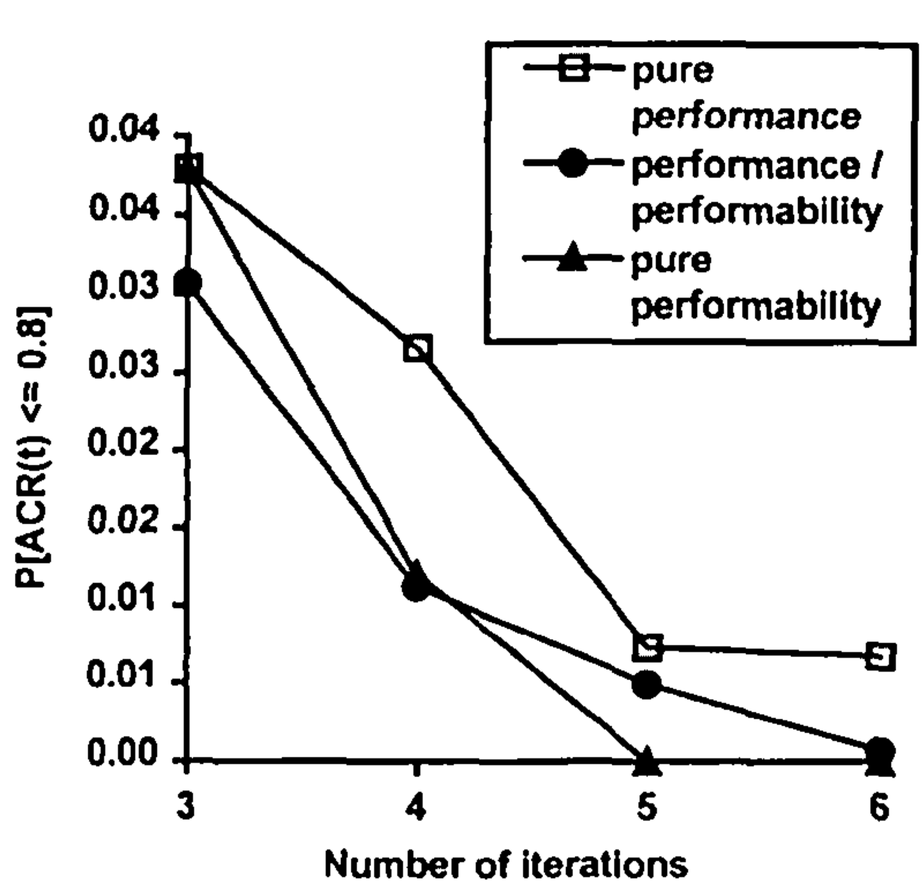


Figure 5.30(b) - Point-performability of networks for iterations 3 to 6

Including the intermediate states of failure does not affect the pure performance approach to design, where performance in states of failure is not required. However, as the rates of partial failure, and repair, are taken into consideration, the overall performability distribution is changed. As far as the performability-based approaches are concerned, allowing for partial failure has a direct impact on the computation involved in evaluation of the performability of the possible topologies at each iteration. This is due to the increase in the size of the state space. A network of m links is now modelled by a Markov chain of $(2m + 1)$ states, whereas previously it consisted of only $(m + 1)$ states. The size of the state space has a direct bearing on the recursive evaluation of the term $\Omega[n, k]$ in Equation (3.5), in Chapter 3. Furthermore, evaluation of the performability index needs to take this additional state into consideration.

Nonetheless, the benefits of using performability-based approaches in design are clear. As shown in Figures 5.29, 5.30(a), and (b), these approaches generate a more fault-tolerant network, and faster than the conventional design technique.

5.14 Conclusions

In this chapter, the capability of the proposed design technique has been investigated, and compared with a conventional approach to design. The design method consists of three stages, where the first two stages are used to design an initial minimally 2-connected network. The third stage, the *fault-tolerance augmentation* stage, is most important. The steps involved consist of iteratively selecting a state for optimisation, deciding where to apply possible augmentation, and using a performability-based descent factor to choose the best topology. The various aspects of design have been investigated.

Firstly, the performability-based descent factor is considered. The descent factor is comprised of a performability measure and the cost of the network. Given a starting topology, an augmentation mechanism is sought that generates a topology such that a maximum decrease in performability is obtained at a minimal increase in cost. Two definitions of cost have been proposed. Both consist of a term which expresses the cost of a network in terms of the cost of its constituent components. The second definition includes

another term which represents the average repair cost expected to be incurred over a period of time. Given that the aim of the proposed technique is to design a network which performs well over a period of time, when it is affected by structural changes such as failure and repair, it is desirable to determine the cost of the network, taking into consideration the occurrence of such changes.

Selecting the state for optimisation at the beginning, and the best topology at the end of an iteration constitute important steps of the third stage of design. In this thesis, it is proposed that optimisation should be carried out in a state of failure associated with the worst performance, or highest performability index. The best topology is chosen with respect to performability. Augmenting in a state of failure ensures that resources are placed in such a way that the network is better equipped to withstand failures. By using performability to choose the topology, the network is chosen, which over a period of time, performs best, taking into consideration how often components fail, and how long the network operates at a degraded level of performance. Results have shown that performability-based approaches outperform conventional design methods at designing fault-tolerant networks, without necessarily being detrimental to the rate at which performance is optimised in the fully operational state. Furthermore, such networks may be obtained using fewer resources than are required by a traditional design approach. The question really boils down to optimal placement of links and the allocation of sufficient capacity to cope with traffic demands in all the states considered. Using a performability index in choice of state does not necessarily lead to any advantage. Results have shown that it does not guarantee that a more fault-tolerant network will be obtained faster.

In this chapter, the design of a 10-node network has been used to illustrate the capability of the proposed technique versus a conventional design method. The difference between these techniques lies in their approach to design, especially in terms of choice of state and choice of topology. By virtue of the way in which these steps are carried out, a performability-based design aims at designing a fault-tolerant network whereas a traditional method neglects the level of performance attained in states of failure. Therefore, the benefits derived are independent of the size of the network being dealt with, and results obtained from design of the 10-node network can be generalised to different networks.

The other aspect of the *fault-tolerance augmentation* stage relates to the augmentation mechanisms. The statement that optimal routing tends to use a single shortest path provided no overload occurs suggested that a capacity augmentation would be favoured to an increase in connectivity. A comparison of these mechanisms and their effects on performance, and performability, has shown that at each iteration of design, choice of the more appropriate mechanism depends on the structure of the network as well as on the traffic scenario being dealt with, regarding distribution of traffic. Both mechanisms have their relative merits.

Finally, the situation where a link is allowed to be in an intermediate state of failure is considered. Including intermediate states of failure does not affect the merits of using a performability-based approach as a design technique for fault-tolerant networks. However, it contributes to an increase in the size of the state space of the performability model, and hence has a direct bearing on the computation involved in evaluation of performability. Prior to that, it requires evaluation of performance in a larger number of states.

The disadvantages of using a performability-based design approach lie in its time, computational, and memory requirements. Performance has to be evaluated for all the allowable states which depends on the number of components constituting the network, and assumptions regarding their failures. Performability then needs to be evaluated for the state space generated. In Chapter 3, techniques of state space reduction were described, and the lumping technique was adopted as a means of reducing the number of states without losing any information regarding the model.

In brief, the use of a performability-based criterion in the iterations of design provides a means of generating topologies which are fault-tolerant. Once minimal reliability requirements have been met, any further augmentation with respect to performability rather than just performance ensures that the network designed has an increased capability of withstanding failures; performance specifications are satisfied when all components are working and degradation upon failures is reduced. Moreover, the network designed is comparable in complexity and cost with one designed according to traditional approaches,

which projects a performability-based design methodology as an attractive option in spite of the intensive computation characterising evaluation of performability.

The performability index with its current definition has not given an accurate indication of which component is more important. A performance-based choice of state has proven to be better in some cases. Application of the index in a context where there are more rewards and hence a finer mapping between performance levels and rewards needs to be investigated. The use of the index may be more prominent then.

The performability-based design method has been developed in the general framework of packet-switched networks, with the routing algorithm and performance measure being chosen for this type of networks. Although the approach has been investigated within this framework, the arguments which constitute the essence of the proposed algorithm, that is optimisation in a state of failure and a performability-based choice of topology, could be extended to different types of networks. In order to be able to apply it to a specific type of network, consideration has to be given to the aspects applicable to the network, such as routing, flow control and choice of the appropriate performance measure. For example, if the design method were to be applied in the context of ATM networks, firstly the routing algorithm has to be changed accordingly. Given that different classes of traffic are being dealt with, the appropriate quality of service figure for each class has to be chosen. The proposed design algorithm has not taken into account issues of flow control. These factors need to be included when applying the design to a particular type of network.

Chapter 6

Performability-based design of clusters

6.1 Introduction

Both network design and performability analysis are plagued by the problem of size. As the number of nodes to be connected increases, the number of potential links between them grows exponentially, with the result that the steps involved in design become more involved, and the performance analysis of the network is made more difficult. Furthermore, with respect to reliability analysis, the number of states in which the network can be, as dictated by failures of its components, increases exponentially with the number of links. As has already been discussed in previous chapters, performability analysis requires evaluation of the performance of the network in all the allowed configurations as part of the actual performability evaluation. In the proposed design technique set out previously, only single link failures have been considered in order to reduce the size of the state space. At the modelling stage, the lumping technique was applied to further decrease the number of states, if possible.

In this chapter, an attempt at dealing with the problems imposed by the size of the network is investigated. The work to be described is at an initial stage of development and a substantial investigation needs to be carried out before any claims can be drawn from it. The approach is to consider the network as a number of interconnected clusters. The gist of the proposed approach is to group the nodes into a number of clusters, design each cluster individually, and then connect the clusters so as to meet the performability specifications.

In addition to reducing the computational complexity of a performability-based design, the approach may be beneficial in deciding where and how to associate network monitors with nodes of the network so as to give a meaningful and rapid indication of changes in performance with the development of faults in the network. Once a network has been designed and is operational, it requires monitoring and maintaining. A usage

network. Adequate measurements of network performance are required so that early steps may be taken, as necessary, to adjust the network design, or expand its overall throughput capacity to meet demand. A maintenance plan is needed to guarantee continuation of service without a break. Corrective action is always required, in some form or another, in order to obtain a good network or prevent component failures, and to maintain overall service standards. Therefore, network monitors are essential to provide such information. If design of a network in clusters is viable, then network monitors can be associated with each cluster, thereby leading to better monitoring of the network and quicker maintenance actions.

In the design of networks in clusters, a number of questions need to be addressed both at the modelling and design stages. The criterion based on which the nodes can be grouped has to be decided upon. The issue that arises in design is how to handle inter-cluster traffic during the design of each cluster such that the performance levels attained for each cluster are maintained as far as possible when the clusters are connected. From the point of view of modelling, this design problem can be thought of as the decomposition and aggregation of a large Markov Reward Model. Therefore, it is necessary to know how to evaluate each cluster, and how to aggregate the results to obtain at the least an estimate of this measure for the whole network. It is also desirable to bound the error involved in this process such that a suitable criterion can be identified at which the design of the clusters can be stopped, and it is anticipated that upon connecting the clusters, performability specifications will be met.

In this chapter, these various issues are investigated. In Section 6.2, the possible criteria which can be used to group the nodes are enumerated. A discussion of possible ways to deal with inter-cluster traffic is given in Sections 6.4 and 6.5. Different scenarios are identified and described in Sections 6.5.1 to 6.5.3. The modelling issues are discussed in Section 6.6 and Section 6.7 consists of a review of the existing decomposition and aggregation techniques, with Section 6.7.1 describing the aggregation method of Schweitzer [SCH90]. The theory of decomposition and the mathematical foundations of the concepts of near complete decomposability are described in Sections 6.7.2 to 6.7.5. In Section 6.8, the concepts are applied to the design of networks. A description of the current network model is given in Section 6.8.1; and the relation between the cluster models and the overall network model is established in Sections 6.8.2 and 6.8.3. Aggregation of results for the

network under steady-state conditions is discussed in Section 6.8.5; and two aggregation techniques for aggregation when a transient analysis is carried out are proposed in Sections 6.8.6.1 and 6.8.6.2. After connecting the clusters, it is desirable to validate the aggregated results against the actual results. The conditions under which this process needs to be carried out are outlined in Section 6.8.7. In Section 6.8.8, the question of establishing a suitable stopping criterion for the design is addressed. The issue of how to estimate this error is treated in Section 6.8.9 and Section 6.9 is devoted to a discussion of the results of designing a network of ten nodes using the clustering technique. The various scenarios are tried and investigated and some conclusions are given in Section 6.10.

6.2 Clustering of nodes

The first question that arises when designing a network in clusters is what criterion needs to be used to group the nodes. A number of different criteria on which this step can be based are outlined below.

1. Distance:- nodes have to be grouped such that the distance between nodes within a cluster is small as compared with the distance in between clusters. This is required because design of each cluster generates a mesh network and cost is dependent upon the length of cables. Also, it is more reasonable to associate a network monitor with nodes which are close together.

2. Size of clusters:- designing in clusters enables a reduction in the amount of computation involved in the process, which is affected by the number of nodes and links in the network. Therefore, the clusters have to be small enough so that computation is not that intensive, but are large enough to avoid the need to design too many clusters. Also, it is likely that the greater the number of clusters, the larger will be the error in estimating the overall performance and performability.

3. Traffic:- another constraint is traffic. To minimise the effects of traffic on performance when the clusters are connected, it is desirable that the nodes are clustered such that traffic demands within a cluster are more intensive than traffic

demands between clusters, that is intra-cluster traffic is greater than inter-cluster traffic.

As has been argued above, one of the reasons for designing a network as a number of interconnected clusters is to be able to associate a network monitor with a set of nodes (a cluster). In this case, the geographical locations of the nodes, and the physical distance between them are more appropriate criteria for grouping the nodes.

6.3 Connecting the nodes

Having decided on a suitable criterion for grouping the nodes, the next question of how to handle inter-cluster traffic during design of any particular cluster needs to be considered. In other words, is the cluster designed prior to the establishment of inter-cluster links? In such a case, what are the assumptions regarding via which nodes inter-cluster traffic enters / leaves each cluster? Positioning of the inter-cluster links has direct implications on how the traffic is distributed within the cluster. One way of minimising the problems arising from consideration of inter-cluster traffic is to group the nodes such that inter-cluster traffic is minimal with respect to intra-cluster traffic so that the design of each cluster is least affected by traffic from other clusters. If clustering does not take traffic into consideration, then the problem can be viewed as a number of scenarios which will be discussed in Sections 6.4 and 6.5.

6.4 Design of clusters after establishment of inter-cluster links

In this approach, the inter-cluster links are established before the design of each cluster begins. The minimal spanning tree connecting the nodes can be derived, and then the nodes are clustered. As a result, a skeleton topology exists for each cluster, as well as for interconnecting the clusters. Consider the example in Figure 6.1.

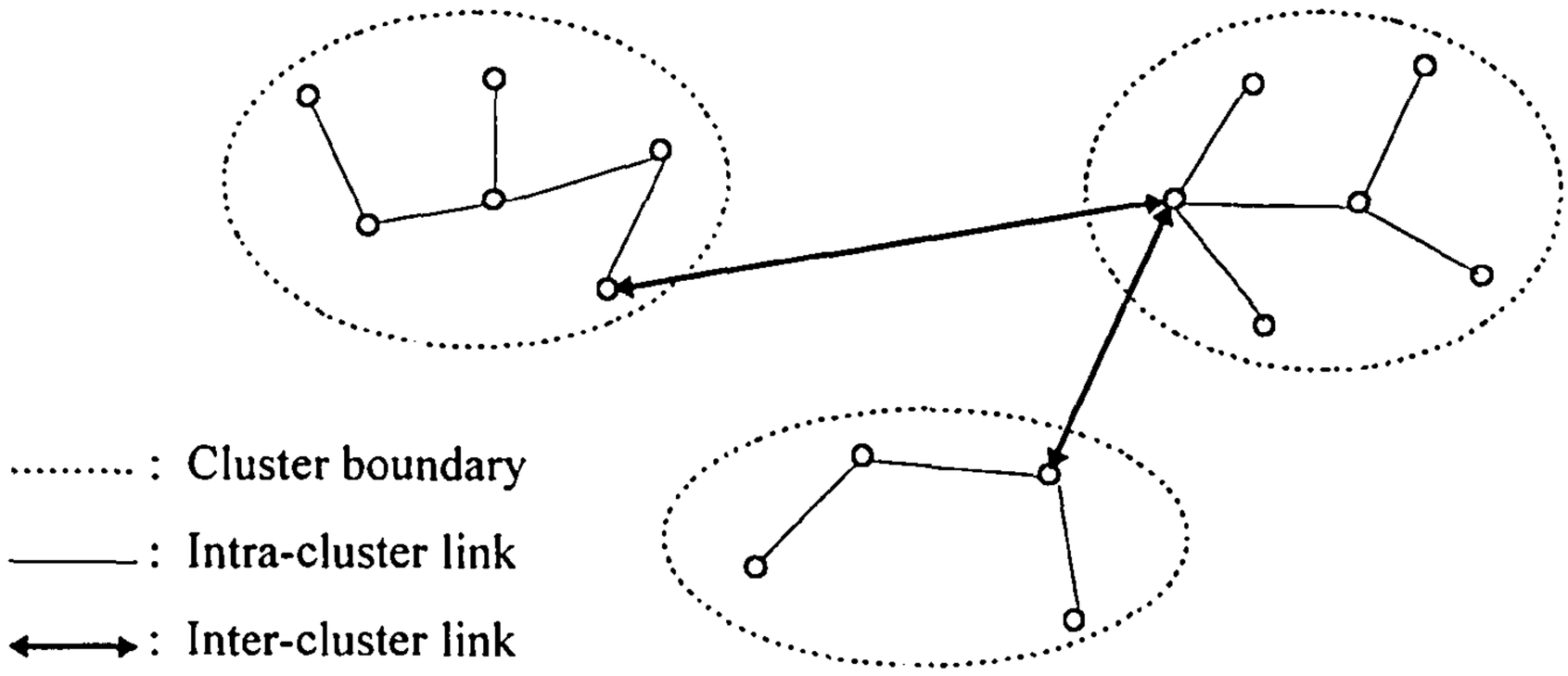


Figure 6.1 - Clusters of nodes

The nodes are clustered in groups as shown by the dotted lines and inter-cluster links are indicated by the solid lines with arrows.

In this case, performability-based design can be applied to each cluster on its own, assuming that the inter-cluster links are highly reliable, and fail very rarely. Having designed each cluster, more inter-cluster links then need to be added to ensure that no disconnection occurs in the event of an inter-cluster link failure. Including extra links at this stage has a direct bearing on the distribution of traffic in each cluster and hence the performance of the network. However, it is anticipated that upon addition of new inter-cluster links, provided there is no reduction in the capacity of the existing links, the capacity of the network is increased, and on application of optimal routing, performance should improve in all the states that can be considered. It is difficult to predict the degree of improvement in performance, which is obtained through this step, during design of each cluster. Therefore, this contributes to the difficulty of deciding on a suitable stopping criterion for the design of each cluster.

6.5 Design of clusters prior to establishment of inter-cluster links

Different scenarios arise from the way in which inter-cluster traffic is assumed to enter/leave the network being designed. Three different scenarios are identified and will be discussed briefly. The term *internal node* will be used to refer to a node within the cluster

being designed whereas an *external node* will be used to refer to a node belonging to other clusters.

6.5.1 *Worst-case* scenario

In this case, from the point of view of each cluster, external clusters are represented by a single node, referred to as the *virtual external node*. Every external cluster may be represented by such a node, or a single *virtual external node* may be used to represent all of them. Furthermore, the cluster being designed is connected to this external node by a single pair of unidirectional links.

A 2-cluster example is used to illustrate this point.

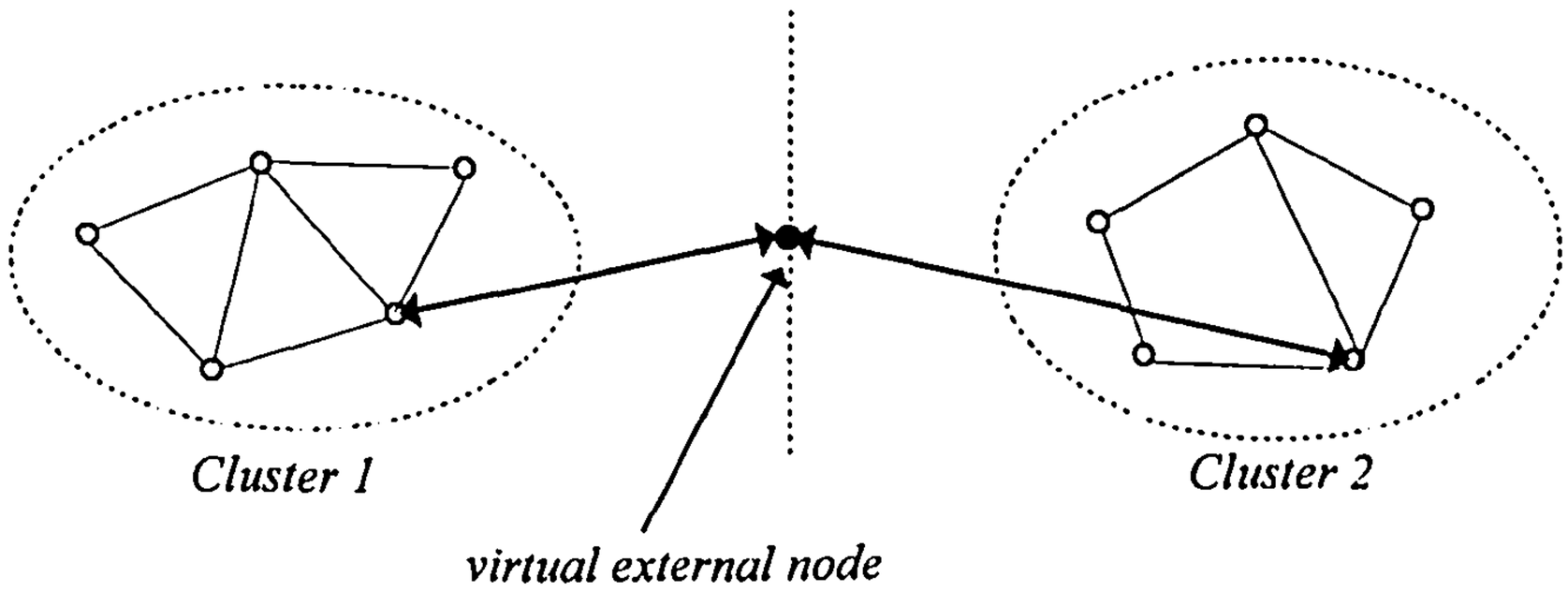


Figure 6.2 - Network configuration in the *worst-case* scenario

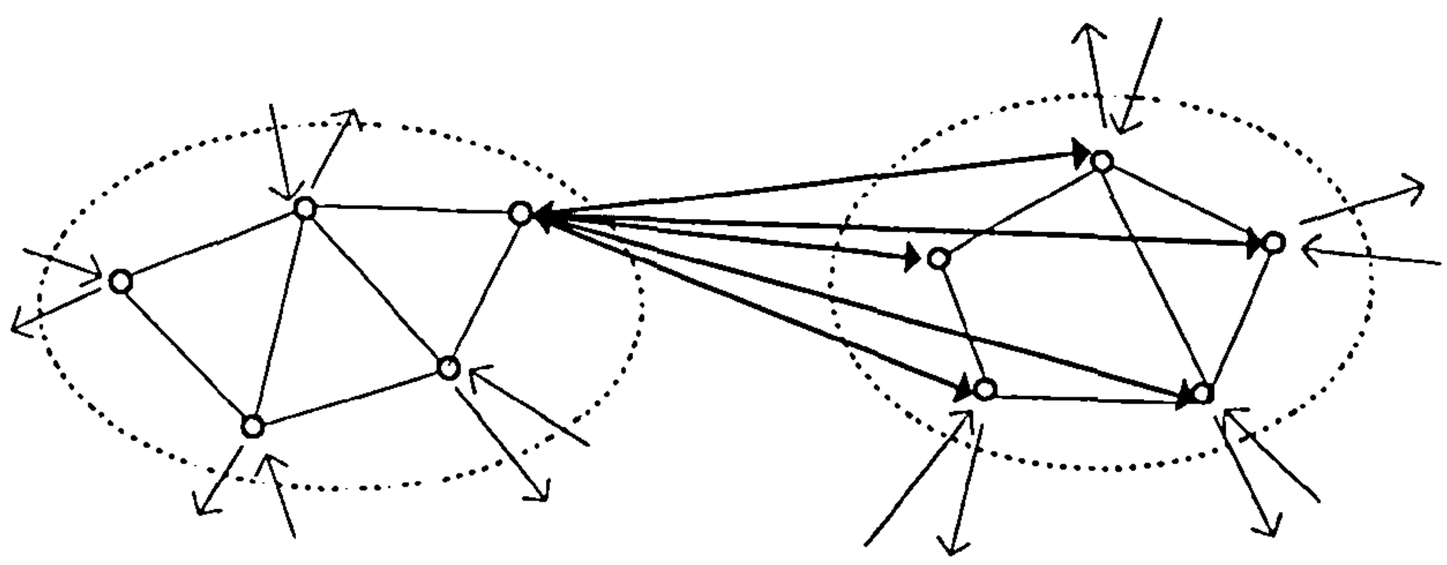
In this approach, design of each cluster has to take into consideration all traffic destined for / arriving from other clusters. As a result, it is expected that sufficient capacity is put into the network for it to cope with inter-cluster traffic. As more inter-cluster links are added, performance is expected to improve.

6.5.2 *Best-case n* scenario

In this scenario, inter-cluster traffic puts minimal load on design of each cluster. Each node within the cluster is assumed to be directly connected to each and every node from other clusters, with which it needs to communicate. Therefore, in essence, each cluster is being designed to handle intra-cluster traffic solely; and nodes are fully connected with respect to

inter-cluster links. Such a topology is expensive so that a less connected inter-cluster topology is desirable. Upon removal of some inter-cluster links, the direct consequence is that clusters have to deal with inter-cluster traffic. Given that they were designed to cope with intra-cluster traffic only, further capacity augmentation is required to prevent performance degradation.

Figure 6.3 is an illustration of the inter-cluster connections in this scenario. Assume that the cluster being designed is cluster 1 so that nodes constituting cluster 2 are treated as external nodes. Although a fully connected mesh topology links the nodes within cluster 1 to external nodes, inter-cluster links between only one node from cluster 1 and other external nodes are shown.



Cluster 1 *Cluster 2*

Figure 6.3 - Network configuration in the *best case n* scenario

6.5.3 *Best-case* 1 scenario

This third scenario falls midway between the previous two. It is similar to the *best case n* approach in the sense that each internal node communicates directly with external clusters for transfer of inter-cluster traffic. Again, inter-cluster traffic puts minimal load on design of each cluster. However, instead of being connected to each and every external node with which it needs to communicate, each internal node is connected to a single virtual external node. Whereas in the *worst-case* scenario, a single virtual external node represented each external cluster, in this case, an external cluster may be represented by a number of nodes.

Owing to the fact that the delay across a high capacity link tends to be less than the sum of delays obtained if the same traffic is distributed among a number of low capacity links, the way in which inter-cluster traffic is dealt with in this approach contributes to a lower delay, that is, a better performance, than the value obtained in the *best case n* scenario.

Figure 6.4 shows the inter-cluster connections. During design of cluster 1, each of its internal nodes is connected to a *virtual external node*.

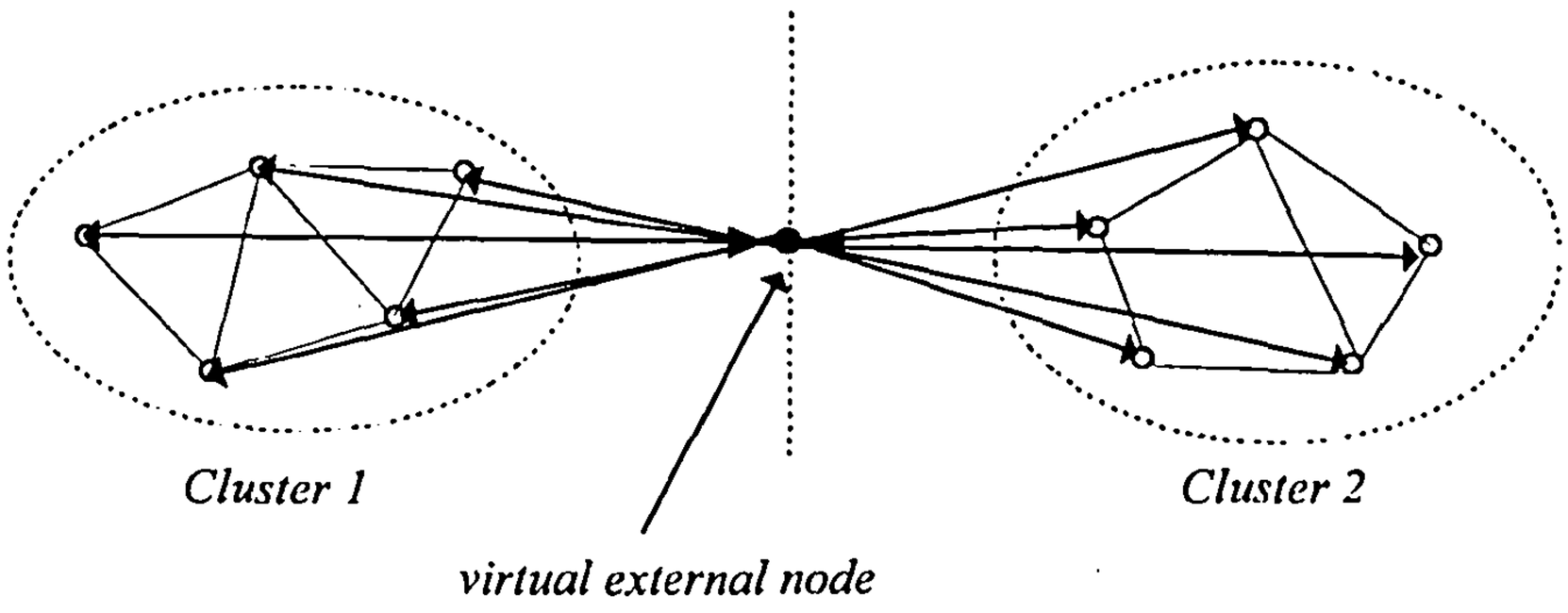


Figure 6.4 - Network configuration in the *best case* 1 scenario

From a capacity point of view, in the *worst-case* scenario, given that at the very start, inter-cluster traffic is assumed to be concentrated into one node, the network designed is equipped with a high capacity. It is designed to cope with a situation where only one pair of unidirectional inter-cluster links exists between two clusters, and deliver a good performance level. As a result, the network is expected to have a high overall capacity and be expensive.

In the two other cases, the cluster itself is being designed to handle intra-cluster traffic only. Therefore, more capacity is required within a cluster in order for it to be able to maintain its performance when faced with the event of an inter-cluster link failure. The network, being of less capacity, is less costly.

Two feasible directions exist if design is based on any of these scenarios. Starting from a network designed according to the *worst-case* scenario, as more inter-cluster links

are added, the capacity of intra-cluster links may be reduced, provided the performance delivered is still acceptable. On the other side, starting from the *best-case* scenarios, more capacity needs to be incorporated into each cluster for it to cope with failure of inter-cluster links.

In consideration of all these different scenarios, the underlying question remains where to stop design of each cluster. At what stage can design of each cluster be stopped such that upon joining the clusters, performance and performability specifications are satisfied within reasonable limits without requiring much effort into inter-cluster design? Such questions depend on how optimal routing would distribute traffic as clusters are connected. What makes these questions even harder to answer is the fact that performability evaluation requires consideration of the network in different failure configurations as well as the fully operational state. It is difficult to anticipate, in a large network, how performance would vary as a result of a redistribution of traffic when more inter-cluster links are added, or are removed, depending on the approach being adopted.

Therefore, in the investigation of how to approach design of clusters, it will be assumed that inter-cluster links are highly reliable such that the possibility that they may fail can be neglected. Such an assumption is made only to simplify the analysis of how performance and performability of the individual clusters relate to the performance and performability of the overall network. It avoids the problems of estimating the variations in performance when there are more inter-cluster links. In the remaining sections, design is assumed to proceed according to the first approach whereby the minimal spanning tree connecting the nodes is first derived before clustering the nodes.

6.6 Modelling issues

On the modelling front, the problem is viewed as a decomposition of a large Markov chain into subchains, finding the measure of interest for each subchain, and then aggregation of these results to obtain a result for the overall system.

6.7 Review of existing techniques

Research into techniques to deal with large-scale systems, which are modelled by large Markov chains, has received considerable attention over the years. Approximate methods are required to overcome the computational limitations imposed by the size of the state space. For example, exact analysis of closed queueing networks can be too costly as the number of states increases, thereby contributing to the intricacy of the model. Reliability models of complex systems may possess an excessively large number of states, which makes the solution computationally intractable. Different types of method exist which attempt to render models easier to solve mathematically or computationally. Iterative methods [DES84], and diffusion methods [HAR92], have been applied to the solution of queueing systems. Commonly used techniques in the solution of Markov processes are the techniques of decomposition and aggregation. Such approaches are particularly suitable in reliability analysis and will be discussed in more detail.

The concepts of decomposition and aggregation have been applied to the solution of queueing systems, as well as to reliability analysis [TRI84]. Most techniques rely on the concepts of near complete decomposability and its associated aggregation technique. The process of aggregation as viewed by Schweitzer [SCH84] [SCH90] and the technique of decomposition explained by Courtois [COU77] will be reviewed in the next section.

6.7.1 Aggregation of large Markov Chains [SCH90]

In this section, the main steps constituting the technique of aggregation and disaggregation of large Markov Chains as explained by Schweitzer [SCH90] will be reviewed. The algorithm is described as it is applied to compute the equilibrium distribution $\pi \equiv \{\pi_i, i \in S\}$ for a discrete-time, time-homogeneous Markov Chain, defined on the finite state space $S = \{1, 2, \ldots, n\}$ where n is large. It is governed by the transition matrix $\mathbf{P} = [P_{ij}]$ which is assumed to be ergodic so that all states communicate and are aperiodic. The equation to be solved is

$$\pi^T = \pi^T \mathbf{P}, \quad \sum_{i \in S} \pi_i = 1 \tag{6.1}$$

The aggregation/disaggregation process involves five key steps:

1. *Partitioning* of the state space into $\bar{n}$ blocks, where usually $\bar{n} << n$.

$$S = S(1) + S(2) + \ldots + S(\bar{n}) \tag{6.2}$$

2. Defining the *aggregate probabilities* for each block $S(\alpha)$:

$$\bar{\pi}_\alpha \equiv \sum_{i \in S(\alpha)} \pi_i = \Pr[in\ S(\alpha)], \quad 1 \le \alpha \le \bar{n} \tag{6.3}$$

3. Specifying the *aggregate equations*:

$$\bar{\pi}_\alpha = \sum_{\beta=1}^{\bar{n}} \bar{\pi}_\beta \bar{\mathbf{P}}_{\beta\alpha} \qquad 1 \le \alpha \le \bar{n} \tag{6.4}$$

$$\sum_{\alpha=1}^{\bar{n}} \bar{\pi}_\alpha = 1 \tag{6.5}$$

where the aggregate transition probability from block $S(\beta)$ to block $S(\alpha)$ is given by

$$\bar{\mathbf{P}}_{\beta\alpha} \equiv \frac{\sum_{j \in S(\beta)} \sum_{i \in S(\alpha)} \pi_j P_{ji}}{\sum_{k \in S(\beta)} \pi_k} \qquad 1 \le \alpha, \beta \le \bar{n} \tag{6.6}$$

4. Specifying the *disaggregation equations* for each block $S(\alpha)$

$$\pi_i = \sum_{\beta=1}^{\bar{n}} \bar{\pi}_\beta P_{\beta i}^{\#} \qquad i \in S(\alpha) \tag{6.7}$$

or

$$\pi_i - \sum_{j \in S(\alpha)} \pi_j P_{ji} = \sum_{\substack{\beta=1 \\ \beta \ne \alpha}}^{\bar{n}} \bar{\pi}_\beta P_{\beta i}^{\#} \qquad i \in S(\alpha) \tag{6.8}$$

where the disaggregate transition probability from block $S(\beta)$ to state i is given by

$$P_{\beta i}^{\#} \equiv \frac{\sum_{j \in S(\beta)} \pi_j P_{ji}}{\sum_{k \in S(\beta)} \pi_k} \qquad 1 \le \beta \le \bar{n}, \quad i \in S. \tag{6.9}$$

5. Devising an *iterative scheme* to solve the set of equations (6.3) to (6.9) for the unknowns π_i, $\bar{\pi}_\alpha$, $\bar{P}_{\beta\alpha}$, and $P_{\beta i}^{\#}$.

6.7.2 Theory of decomposition

The technique of decomposition is a widely used approximate method where its essence relies on the idea that for many large systems, the variables can be clustered into a number of groups such that the interactions among the groups may be ignored during analysis of each group, and conversely these interactions can be studied without reference to the interactions within a group. Therefore, each group can be evaluated individually, and the results from each group can be aggregated in a suitable way depending on the interactions between the groups to generate an approximate evaluation of the overall system. The aggregation of variables gives good results under two different types of conditions: firstly when interactions between groups of variables are independent from interactions within groups, and secondly when the variables are functions of variables of the same group, that is the interactions between groups of variables are null [COU77]. Such a system is completely decomposable in the sense that it consists of independent systems, each of which can be analysed separately.

Simon and Ando [SIM61] have investigated the process of aggregation of variables in systems where interactions between groups of variables are non-null and arbitrary, but weak compared with interactions within the groups. These types of system are referred to as being *nearly completely decomposable* (NCD). Simon and Ando showed that in these systems, aggregation of variables must separate the analysis of the short-run dynamics from that of the long-run dynamics. In the short run, the system behaves approximately as if it were completely decomposable, each subsystem approaching its local equilibrium approximately independently of the others. When this short term period is over, these local equilibrium states remain approximately maintained in each subsystem during the period when the system, as a whole, evolves towards a global equilibrium under the influence of the weak interactions among subsystems [COU77].

6.7.3 Mathematical description of NCD [COU77]

The principles underlying the concepts of *near complete decomposability* and its associated *aggregation* technique for stochastic systems as explained by Courtois [COU77] will now be discussed.

Consider a stochastic system governed by the equation

$$\tilde{\mathbf{y}}(t+1) = \tilde{\mathbf{y}}(t)\mathbf{A} \tag{6.10}$$

where $\tilde{\mathbf{y}}(t)$ is a row probability vector

and $\mathbf{A}$ is a stochastic matrix of order n.

The system consists of n states; $y_l(t)$ is the unconditional probability of the system being in state l, $l = 1,\ldots, n$, at time t; and the element a_{kl} of matrix $\mathbf{A}$ is the conditional probability that the system is in state l at time t given that it was in state k at time t-1.

Consider the system governed by

$$\tilde{\mathbf{y}}^{\bullet}(t+1) = \tilde{\mathbf{y}}^{\bullet}(t)\mathbf{A}^{\bullet} \tag{6.11}$$

where $\mathbf{A}^{\bullet}$ is a stochastic matrix of order n and is given by

$$\mathbf{A}^{\bullet} = \begin{bmatrix} \mathbf{A}_1^{\bullet} & & & \\ & \ddots & & \\ & & \mathbf{A}_I^{\bullet} & \\ & & & \ddots \\ & & & & \mathbf{A}_N^{\bullet} \end{bmatrix}$$

As shown, $\mathbf{A}^{\bullet}$ is in block diagonal form where the $\mathbf{A}_I^{\bullet}$ are square submatrices, and the remaining elements are equal to zero.

Let $n(I)$ denote the order of $\mathbf{A}_I^{\bullet}$ so that $n = \sum_{I=1}^{N} n(I)$. (6.12)

The vector $\tilde{\mathbf{y}}^{\bullet}(t)$ is given by

$$\tilde{\mathbf{y}}^{\bullet}(t) = \left[y_k^{\bullet}(t)\right] = \left[\left[y_{i_1}^{\bullet}(t)\right] \ldots \left[y_{i_J}^{\bullet}(t)\right] \ldots \left[y_{i_N}^{\bullet}(t)\right]\right],$$

where $\left[y_{i_I}^{\bullet}(t)\right]$ is a vector of elements of $\left[y_k^{\bullet}(t)\right]$ so that if

$$y_{i_I}^{\bullet}(t) = y_k^{\bullet}(t)$$

then

$$k = \sum_{J=1}^{I-1} n(J) + i \tag{6.13}$$

The matrix $\mathbf{A}^*$ is said to be *completely decomposable* whereas the matrix $\mathbf{A}$ is referred to as being *nearly completely decomposable* if it can be written in the form

$$\mathbf{A} = \mathbf{A}^* + \varepsilon\mathbf{C}, \tag{6.14}$$

where $\mathbf{C}$ is a square matrix of the same order as $\mathbf{A}^*$, which has the property of keeping both $\mathbf{A}$ and $\mathbf{A}^*$ stochastic,

and ε is a real positive number, small compared to the elements of $\mathbf{A}^*$.

The row and column partition of $\mathbf{A}^*$ are applied to $\mathbf{A}$ and $\mathbf{C}$ such that $\mathbf{A}_{IJ}$ is used to denote the submatrix of $\mathbf{A}$ at the intersection of the I^{th} set of rows and the J^{th} set of columns, and $a_{i_I j_J}$ refers to the element at the intersection of the i^{th} row and the j^{th} column of $\mathbf{A}_{IJ}$.

In order to keep matrices $\mathbf{A}$ and $\mathbf{A}_I^*$, $I = 1, \ldots, N$, stochastic, the row sums of $\mathbf{C}$ have to be equal to zero. Nonpositive values are chosen for the elements $c_{i_I k_I}$ and nonnegative values are chosen for all elements $c_{i_I k_J}$, $I \neq J$, so that for each row i_I

$$\sum_{k=1}^{n(I)} c_{i_I k_I} = -\sum_{J=1, J\neq I}^{N} \sum_{j=1}^{n(J)} c_{i_I j_J} \tag{6.15}$$

Moreover, ε and $\mathbf{C}$ are chosen such that for all rows i_I

$$\varepsilon \sum_{J=1, J\neq I}^{N} \sum_{j=1}^{n(J)} c_{i_I j_J} = \sum_{J=1, J\neq I}^{N} \sum_{j=1}^{n(J)} a_{i_I j_J} \tag{6.16}$$

and

$$\varepsilon = \max_{i_I} \left(\sum_{J=1, J\neq I}^{N} \sum_{j=1}^{n(J)} a_{i_I j_J} \right). \tag{6.17}$$

Thus,

$$\max_{i_I}\left(\sum_{J=1,J\neq I}^{N}\sum_{j=1}^{n(J)}c_{i_I j_J}\right)=-\max_{i_I}\left(\sum_{k=1}^{n(I)}c_{i_I k_I}\right)=1 \tag{6.18}$$

and $$\left|c_{i_I j_J}\right|\leq 1 \text{ for all } i_I,\ j_J\,. \tag{6.19}$$

ε is referred to as the maximum degree of coupling between subsystems $\mathbf{A}_{II}$.

Furthermore, Courtois [COU77] has formulated the following sufficient condition for near complete decomposability:

$$\varepsilon < \frac{1}{2}\left[1-\max_{I}\left|\lambda^{*}(2_I)\right|\right] \tag{6.20}$$

where $\lambda^{*}(2_I)$ is the second largest eigenvalue of diagonal submatrix *I*.

Equation (6.14) makes possible the expression of a nearly completely decomposable system in terms of a completely decomposable one. The decomposition stage then involves the analysis of the system of order *n* in terms of *N* small $n(I)$ x $n(I)$ independent subsystems. Aggregation of these results for the *N* x *N* system is required to yield an estimate of the results of interest for the overall system.

6.7.4 The process of aggregation

Courtois [COU77] summarises the problem of aggregation as follows. The feasibility of aggregation in a linear *n* x *n* system of the form

$$\tilde{\mathbf{y}}(t+1)=\tilde{\mathbf{y}}(t)\mathbf{A} \tag{6.21}$$

depends on the existence of

1. *N aggregating* functions

$$Y_I(t)=\mathcal{A}_I(y_i(t)), \qquad I=1,\ldots,N, \tag{6.22}$$

2. *n disaggregating* functions

$$y_i(t) = \mathcal{D}_i(Y_I(t)), \qquad i = 1, \ldots, n, \tag{6.23}$$

3. a new N x N system of relations

$$\tilde{\mathbf{Y}}(t+1) = \tilde{\mathbf{Y}}(t)\mathbf{B}, \tag{6.24}$$

the elements of **B** being functions of those of **A**.

It is desirable that N is much smaller than n, while the elements of **B** and the disaggregating functions are rather easy functions to compute. The error involved in this process also has to be acceptable in the sense that the N x N system **B** and the disaggregating functions have to be such that the time behaviour of $y_i(t)$, obtained using equations (6.23) and (6.24) is an acceptable approximation of the time behaviour of the system as defined by equation (6.21). The way in which this process is carried out depends on the measure being sought and the type of analysis the system is being subjected to, that is, a steady-state analysis or a transient analysis. In the next section, aggregation of results under steady-state conditions is discussed.

6.7.5 Aggregation in the steady state [COU77]

Let $y_{i_I}(t)$ be the probability of being in state i of cluster or diagonal matrix I. According to Simon and Ando, there exists a time T after which the probabilities $y_{i_I}(t)$ of each group I move in proportion to each other, keeping approximately the same ratio as the elements of the steady-state vector of $\mathbf{A}_I^*$, denoted by $\mathbf{v}^*(1_I)$, the left eigenvector corresponding to the first eigenvalue of submatrix I.

Therefore, if the aggregation equations are defined as

$$Y_I(t) = \sum_{i=1}^{n(I)} y_{i_I}(t) \tag{6.25}$$

for $t > T$,

$$\frac{y_{i_I}(t)}{Y_I(t)} \approx v_{i_I}^*(1_I) \tag{6.26}$$

Furthermore, the probability that the system will be at time t+1 in any one state j_J , $j = 1, \ldots, n(J)$, of cluster J, on condition that it was in any one state i_I, $i = 1, \ldots, n(I)$, of group I at time t, is given by

$$b_{IJ}(t+1) = (Y_I(t))^{-1} \sum_{i=1}^{n(I)} y_{i_I}(t) \sum_{j=1}^{n(J)} a_{i_I j_J} \tag{6.27}$$

For $t > T$,

$$b_{IJ}(t) \approx b_{IJ} = \sum_{i=1}^{n(I)} v_{i_I}^{\bullet}(1_I) \sum_{j=1}^{n(J)} a_{i_I j_J} \tag{6.28}$$

The time-independent matrix $\mathbf{B} = [\, b_{IJ} \,]$ is stochastic and the elements, say $X_I(1)$, of its steady-state vector $\mathbf{X}(1)$ are good approximations, for $t > T$, to the probabilities $Y_I(t)$ of being in any one state of the subset I. $\mathbf{X}(1)$ is obtained by solving the system

$$\mathbf{X}(1)(\mathbf{B} - \mathbf{I}_N) = \mathbf{0}\,. \tag{6.29}$$

The disaggregating equations as given by

$$x_{i_I}(1) = X_I(1) v_{i_I}^{\bullet}(1_I), \qquad i = 1,\ldots,n(I), \quad I = 1,\ldots,N \tag{6.30}$$

are good approximations to the steady-state probabilities $v_{i_I}(1_1)$ of being in any particular state i_I.

Through this process, the resolution of the system of order n given by

$$v(1_1)[\mathbf{A} - \mathbf{I}_n] = 0 \tag{6.31}$$

has been reduced to the resolution of N small $n(I)$ x $n(I)$ independent systems

$$v^{\bullet}(1_I)[\mathbf{A}_I^{\bullet} - \mathbf{I}_{n(I)}] = 0 \tag{6.32}$$

and of the N x N system given by equation (6.29).

According to Courtois [COU75] [COU77], this classical process of aggregation yields an approximation of accuracy ε to the eigenvector of the original matrix associated with the dominant eigenvalue, which corresponds to the steady-state probability distribution.

6.8 Application of decomposition and aggregation to the network model

In performability-based design of networks, the problem consists of

1. decomposing the given set of nodes into a number of clusters,
2. designing and evaluating each cluster with respect to performability, and
3. aggregating these individual assessments to yield an estimate of the overall network's performability. The issue of the accuracy of the estimate needs to be considered.

In terms of specifications, the problem can be viewed in two ways: specifications are usually given for the whole network. Firstly, when design is done in clusters, the specifications can be translated into specifications for each cluster. These establish a suitable stopping criterion for design of each cluster. These stopping criteria have to be identified such that on merging the clusters, the whole network will be able to meet the overall specifications. The second way of approaching the problem is to estimate the error involved in the aggregation process and hence translate the specifications into specifications for the aggregated network. In other words, if the aggregated results are within certain bounds of the objective performability requirements, the network is judged to be sufficiently equipped to meet the specifications.

Therefore, the questions that need to be addressed are how to evaluate the sub-networks, and more importantly what function to use to aggregate the results. What is the error involved in this process, which should lead to establishment of a suitable stopping criterion for design of the individual clusters? In order to be able to do so, it is necessary to understand how the process of decomposition and aggregation is applied to the network model.

6.8.1 Description of the current network model

Performability analysis and evaluation has relied on the description of the network, or system of components, in terms of a Markov Reward Model. As explained in Chapter 3, this model is comprised of an underlying Markov Process augmented by a reward structure. The Markov Process represents the failure and repair actions which affect the system whereas

the reward assigned to each state may give an indication of the performance of the system while operating in that state.

The system, defined on the state space S = $\{a_i : i = 0, \ldots, M\}$ is characterised by a generator matrix **Q**, an initial state probability distribution $\pi(0)$, and a reward structure ρ. An entry q_{ij}, from matrix **Q**, corresponds to the rate of a transition occurring from state a_i to state a_j whereas entries ρ_i and $\pi_i(0)$ are equal to the reward attached to, and the probability that the system starts in state a_i respectively. Using randomisation, the Continuous Time Markov Process can be viewed as a Discrete Time Markov Chain subordinated to a Poisson Process. The Discrete Time Markov Chain is governed by the stochastic transition matrix **P** where $\mathbf{P} = \mathbf{Q}/\Lambda + \mathbf{I}$, and Λ is chosen such that $\Lambda \geq \max. \{q_i\}$ for $1 \leq i \leq M$. A typical transition matrix is in the following form.

$$\mathbf{P} = \begin{bmatrix} 1-\Sigma q_{0i}/\Lambda & q_{01}/\Lambda & q_{02}/\Lambda & \cdots & q_{0M}/\Lambda \\ q_{10}/\Lambda & 1-\Sigma q_{1i}/\Lambda & q_{12}/\Lambda & \cdots & q_{1M}/\Lambda \\ q_{20}/\Lambda & q_{21}/\Lambda & 1-\Sigma q_{2i}/\Lambda & \cdots & q_{2M}/\Lambda \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{M0}/\Lambda & q_{M1}/\Lambda & q_{M2}/\Lambda & \cdots & 1-\Sigma q_{Mi}/\Lambda \end{bmatrix}$$

The reward structure and initial state probability distribution are shown below as row vectors:

Reward structure $\rho = [\rho_0 \quad \rho_1 \quad \rho_2 \quad \cdots \quad \rho_M]$

Initial state probability distribution $\pi(0) = [\pi_0 \quad \pi_1 \quad \pi_2 \quad \cdots \quad \pi_M]$.

In this thesis, one of the assumptions has been that only single link failures are allowed. In this case, the transition probability matrix is in the form

$$\mathbf{P} = \begin{bmatrix} 1-\Sigma\lambda_i/\Lambda & \lambda_1/\Lambda & \lambda_2/\Lambda & \cdots & \lambda_M/\Lambda \\ \mu_1/\Lambda & 1-\mu_1/\Lambda & 0 & \cdots & 0 \\ \mu_2/\Lambda & 0 & 1-\mu_2/\Lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mu_M/\Lambda & 0 & 0 & \cdots & 1-\mu_M/\Lambda \end{bmatrix}$$

where the elements of the first row, first column and the diagonal elements are the only non-zero entries.

Decomposition of the network entails a decomposition of the reliability model described by the transition matrix and initial state probability distribution, and the performance model represented by the reward structure. Each cluster needs to be characterised by these parameters. The manner in which decomposition and aggregation are applied to this model is explained in the following sections. The following discussion with respect to the difficulty of relating the clusters' models to the overall model assumes only single failures. The same drawback is encountered in a system or network where multiple failures and partial repair are allowed.

6.8.2 Relation between the overall network and clusters' reliability models

At first glance, the transition matrix governing the overall network model appears not to be in a decomposable form. More importantly, given the transition matrices associated with the cluster models, it is difficult to establish the relation with the overall model, and hence it is not apparent how to aggregate results obtained for each cluster to generate an overall result. The difficulty arises from the fact that each cluster is associated with a fully operational state and these states are aggregated into a single state in the overall network model. Yet, a failure state in a cluster model has a single matching state in the overall network model.

To illustrate this, consider the following example. A system of four components is decomposed into two clusters of two components each. The overall network model is as shown in Figure 6.5.

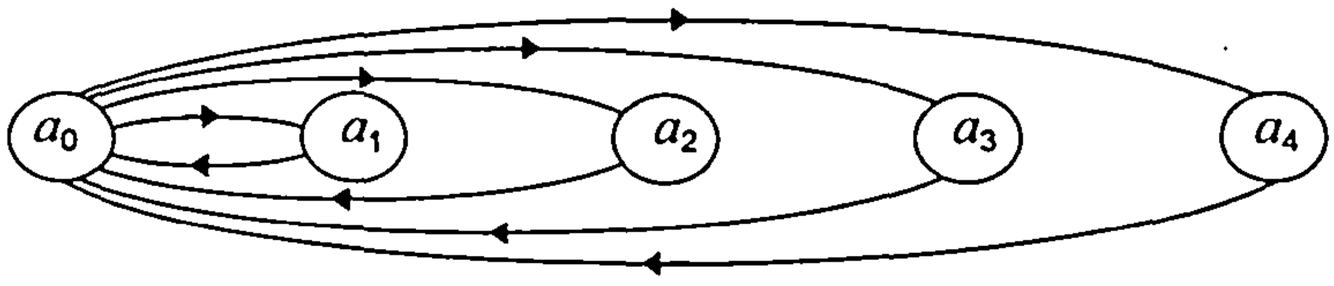


Figure 6.5 - Overall network model

The Markov Reward Models corresponding to the clusters are as shown in Figures 6.6(a) and 6.6(b).

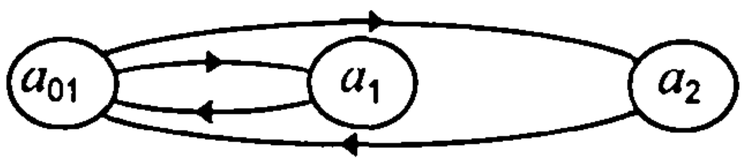


Figure 6.6(a) - MRM for cluster 1

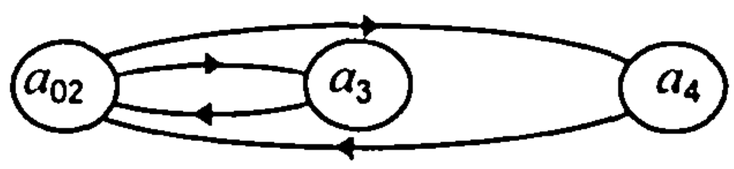


Figure 6.6(b) - MRM for cluster 2

As can be seen, states a_1 to a_4, corresponding to failures of components 1 to 4 respectively, each have a matching state in the overall model. However, the fully operational states for the two cluster models, states a_{01} and a_{02}, are represented by a single state, state a_0. This is further illustrated in the transition matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ corresponding to the clusters, and $\mathbf{P}_{ov}$ for the whole system.

$$\mathbf{P}_1 = \begin{bmatrix} 1-\Sigma\lambda_i/\Lambda & \lambda_1/\Lambda & \lambda_2/\Lambda \\ \mu_1/\Lambda & 1-\mu_1/\Lambda & 0 \\ \mu_2/\Lambda & 0 & 1-\mu_2/\Lambda \end{bmatrix}$$

$$\mathbf{P}_2 = \begin{bmatrix} 1-\Sigma\lambda_i/\Lambda & \lambda_3/\Lambda & \lambda_4/\Lambda \\ \mu_3/\Lambda & 1-\mu_3/\Lambda & 0 \\ \mu_4/\Lambda & 0 & 1-\mu_4/\Lambda \end{bmatrix}$$

$$\mathbf{P}_{ov} = \begin{bmatrix} 1-\Sigma\lambda_i/\Lambda & \lambda_1/\Lambda & \lambda_2/\Lambda & \lambda_3/\Lambda & \lambda_4/\Lambda \\ \mu_1/\Lambda & 1-\mu_1/\Lambda & 0 & 0 & 0 \\ \mu_2/\Lambda & 0 & 1-\mu_2/\Lambda & 0 & 0 \\ \mu_3/\Lambda & 0 & 0 & 1-\mu_3/\Lambda & 0 \\ \mu_4/\Lambda & 0 & 0 & 0 & 1-\mu_4/\Lambda \end{bmatrix}$$

From the transition matrices, it is not easy to see how to aggregate matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ to generate matrix $\mathbf{P}_{ov}$.

In order to be able to apply Courtois's principles of decomposition and aggregation to the network model, it is necessary to apply the following transformation to the model and its transition matrix. The model is expanded such that it is defined on the state space $T = \{ b_i : i = 0, \ldots, D\text{-}1\}$ where $D = M + L$; M is the number of links in the network, and L is the number of clusters in which it is to be considered. In essence, in the new expanded model, extra states are included such that each cluster is associated with a fully operational state.

The original model and the corresponding one resulting from the transformation are as shown in Figures 6.7 and 6.8. Each failure state in the original model is represented by a single state in the new model, for example, state a_1 is mapped on to state b_1. However, the fully operational state is represented by x number of states, where x is the number of clusters in which the network is being considered. Therefore, state a_0 is mapped on to states b_0, b_2, and $b_{D\text{-}3}$, among other states in Figure 6.8.

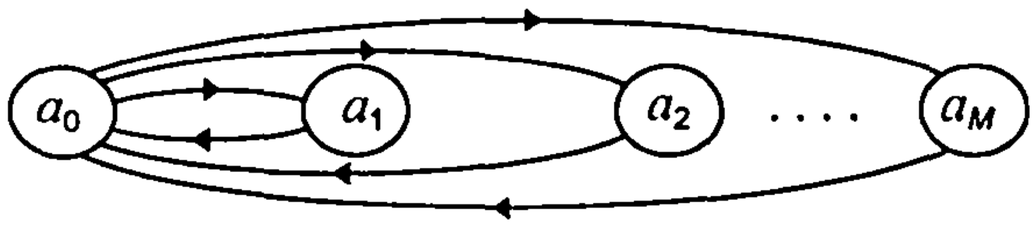


Figure 6.7 - Original network model

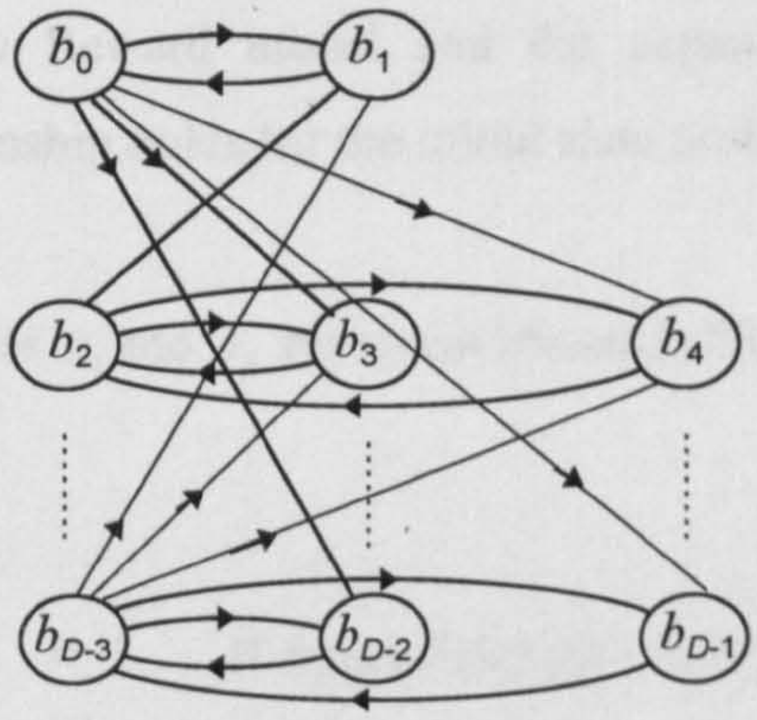


Figure 6.8 - Expanded network model.

The new transition matrix **P'** and its associated initial state probability distribution $\boldsymbol{\pi}'$ are as shown below. The shaded area indicates the states representing a fully operational configuration.

$$\mathbf{P}' = \left[\begin{array}{cc:c:ccc:c:cc}
1-\Sigma\lambda_i/\Lambda & \lambda_1/\Lambda & 0 & \lambda_2/\Lambda & \lambda_3/\Lambda & \cdots & \lambda_{M-1}/\Lambda & \lambda_M/\Lambda \\
\mu_1/\Lambda & 1-\mu_1/\Lambda & 0 & 0 & 0 & \cdots & 0 & 0 \\
\hdashline
0 & \lambda_1/\Lambda & 1-\Sigma\lambda_i/\Lambda & \lambda_2/\Lambda & \lambda_3/\Lambda & \cdots & \lambda_{M-1}/\Lambda & \lambda_M/\Lambda \\
0 & 0 & \mu_2/\Lambda & 1-\mu_2/\Lambda & 0 & \cdots & 0 & 0 \\
0 & 0 & \mu_3/\Lambda & 0 & 1-\mu_3/\Lambda & \cdots & 0 & 0 \\
\hdashline
0 & \lambda_1/\Lambda & 0 & \lambda_2/\Lambda & \lambda_3/\Lambda & \cdots & \lambda_{M-1}/\Lambda & \lambda_M/\Lambda \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1-\mu_M/\Lambda
\end{array}\right]$$

$$\boldsymbol{\pi}' = \left[\begin{array}{cc:c:ccc:c:cc}
\pi_{b_1} & \pi_{b_2} & \pi_{b_3} & \pi_{b_4} & \pi_{b_5} & \cdots & \pi_{b_{D-2}} & \pi_{b_{D-1}}
\end{array}\right]$$

The dotted lines show how the matrix can be partitioned into submatrices where the diagonal submatrices correspond to the transition matrices associated with the clusters. The partitions in the initial state probability distribution yield subvectors to be associated with each submatrix.

The original Markov Reward model and the expanded model are equivalent provided the following relationship holds for the initial state probability distribution.

$$\pi_{a_i} = \pi_{b_j} \qquad \text{if states } a_i \text{ and } b_j \text{ represent the same failure state} \qquad (6.33)$$

$$\pi_{a_0} = \sum_{x=0}^{D-1} \pi_{b_x} I(b_x) \qquad (6.34)$$

where $I(b_x) = 1$, if b_x is a fully operational state

$= 0$, otherwise.

Upon expansion of the MRM, the probability of starting in a failure state remains the same; the sum of the probabilities of starting in any of the fully operational states in the new model is equal to the probability of starting in state a_0 in the original model.

In terms of rewards, the following relationship needs to hold:

$$\rho_{b_j} = \rho_{a_0} \qquad \text{if state } b_j \text{ represents a fully operational state} \qquad (6.35)$$

$= \rho_{a_i}$ if states a_i and b_j represent the same failure state

If these conditions are satisfied, the two models are equivalent.

Upon application of this transformation, matrix **P'** is now in a nearly completely decomposable form and can be written in terms of $\mathbf{P}^* + \varepsilon\mathbf{C}$, according to equation (6.14). Both $\mathbf{P}^*$ and **C** are of order D and are as shown below.

$$\mathbf{P}^* = \left[\begin{array}{cc:ccc:ccc}
1-\lambda_1/\Lambda & \lambda_1/\Lambda & 0 & 0 & 0 & \cdots & 0 & 0 \\
\mu_1/\Lambda & 1-\mu_1/\Lambda & 0 & 0 & 0 & \cdots & 0 & 0 \\
\hdashline
0 & 0 & 1-(\lambda_2+\lambda_3)/\Lambda & \lambda_2/\Lambda & \lambda_3/\Lambda & \cdots & 0 & 0 \\
0 & 0 & \mu_2/\Lambda & 1-\mu_2/\Lambda & 0 & \cdots & 0 & 0 \\
0 & 0 & \mu_3/\Lambda & 0 & 1-\mu_3/\Lambda & \cdots & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & \cdots & \lambda_{M-1}/\Lambda & \lambda_M/\Lambda \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1-\mu_M/\Lambda
\end{array}\right]$$

The diagonal elements $P_i^{\bullet}$ are such that for each row i, $P_i^{\bullet} = 1 - \sum_{j=0}^{D-1} P_{ij}^{\bullet}$.

$$\varepsilon\mathbf{C} = \left[\begin{array}{c|cc|ccc|c|cc}
-\Sigma\lambda_i/\Lambda & 0 & 0 & \lambda_2/\Lambda & \lambda_3/\Lambda & \cdots & \lambda_{M-1}/\Lambda & \lambda_M/\Lambda \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
\hline
0 & \lambda_1/\Lambda & -\Sigma\lambda_i/\Lambda & 0 & 0 & \cdots & \lambda_{M-1}/\Lambda & \lambda_M/\Lambda \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
\hline
0 & \lambda_1/\Lambda & 0 & \lambda_2/\Lambda & \lambda_3/\Lambda & \cdots & \lambda_{M-1}/\Lambda & \lambda_M/\Lambda \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0
\end{array}\right]$$

For each row i, the diagonal elements C_i are given by $C_i = -\sum_{j=0}^{D-1} C_{ij}$.

$\mathbf{P}^{\bullet}$ is in block diagonal form, thus fully decomposable such that each of its block diagonal matrices can be manipulated independently. The state probability vector is decomposed into subvectors which are each associated with a diagonal submatrix.

$$\boldsymbol{\pi}' = [[\boldsymbol{\pi}_1][\boldsymbol{\pi}_2]\cdots[\boldsymbol{\pi}_L]]$$

where $[\boldsymbol{\pi}_l]$ is the vector corresponding to diagonal submatrix $\mathbf{P}_l^{\bullet}$.

It should be noted that upon decomposition of the vector, each subvector needs to be normalised so that the sum of its elements is equal to 1.

The relation between the overall network model and the cluster models is clearer. The diagonal matrices correspond to the transition matrices associated with the cluster models.

6.8.3 Relation between the overall network and the clusters' performance model

The relation between the reward structure associated with the overall network and the clusters follows the same relation as the initial probability distribution. The rewards associated with the failure states are the same as they are for the original model whereas the reward attached to any of the states representing a failure-free configuration is equal to the reward associated with the fully operational state in the initial model. If the reward structure for the overall network is as shown below, the reward structures for the clusters correspond to the subvectors identified by the partitioning shown by the dotted lines.

$$\rho' = \left[\rho_{a_0} \quad \rho_{a_1} \,\vdots\, \rho_{a_0} \quad \rho_{a_2} \quad \rho_{a_3} \,\vdots\, \rho_{a_0} \quad \rho_{a_4} \quad \rho_{a_5} \,\vdots\, \rho_{a_0} \quad \cdots \quad \rho_{a_M} \right]$$

$$\rho' = [[\rho_1][\rho_2]\cdots[\rho_L]]$$

where ρ_L corresponds to the reward structure for cluster L.

6.8.4 Design of clusters

Consider a network of M components decomposed into L clusters. For each cluster, its possible configurations as dictated by failures of its links are identified. The performance of the sub-network can be determined in each of these states, leading to construction of a reward structure. In evaluation of performance, the inter-cluster links may, or may not, be taken into consideration. For example, if performance is measured in terms of the delay across the network, the delay across the inter-cluster links may also be included. The performability of the cluster can then be determined.

Having designed and evaluated each cluster with respect to performability, the next step involves aggregating the results to obtain an estimate of the performability of the overall network.

6.8.5 Aggregation of the network model under steady-state conditions

In Section 6.7.5, the process of aggregation has been applied to approximate the steady-state probability distribution associated with a given matrix. In the network context, evaluation of the equilibrium probability distribution leads to evaluation of the mean accumulated reward of the network.

$$\text{Mean accumulated reward} = \pi_{equi} \cdot \rho \tag{6.36}$$

where π_{equi} is the equilibrium probability distribution,

and ρ is the reward vector.

Given that the elements of the steady-state probability distribution of the overall network can be estimated with an accuracy ε, the mean accumulated reward associated with the overall network is therefore estimated with an accuracy of $D\varepsilon\rho$, where D is the number of states in the model. The maximum error expected is equal to $D\varepsilon$, in the case when all elements of the reward vector are equal to 1.0, given that rewards vary between 0 and 1.

6.8.6 Proposed aggregation technique for transient analysis

The question investigated in this section is what function to use to aggregate the performabilities of the individual clusters so as to obtain an approximate measure of the overall network performability when a transient analysis is carried out. Two approaches will be discussed. In the first case, exact results are obtained for the network in its fully decomposable form, and in the second case, attempts are made to generate approximate results for the network in its nearly completely decomposable form.

6.8.6.1 Aggregation for the completely decomposable network

The proposed function for aggregating the measures is as follows:

$$x_{agg}(t) = \sum_{i=1}^{L} x_i(t)\,\Pr_i(0) \tag{6.37}$$

where $x_{agg}(t)$ is the aggregated measure,

$x_i(t)$ is the measure associated with cluster i,

and $\Pr_i(0)$ is the probability that the network starts in a state in cluster i at time equals 0.

The probability of the network starting in cluster i is given by the sum of the probabilities of starting in a state in the model for that particular cluster as given by equation (6.38).

$$\Pr_i(0) = \sum_{j=0}^{D-1} \Pr_{ij}(0) \tag{6.38}$$

where $\Pr_{ij}(0)$ is the probability that the network starts in a state j of cluster i at time equals 0.

Aggregation using equation (6.37) gives exactly the same results as results obtained for the system governed by the transition matrix $\mathbf{P}^*$, that is, the system in its fully decomposable form. The two systems are associated with the same probability distribution. The above equation gives an estimate of the performability measure associated with the nearly decomposable system. Determination of the error involved in this process is not an easy task especially since the analysis is transient and time-dependent.

6.8.6.2 Aggregation for the nearly completely decomposable system

In this approach, the approximate result is obtained after the solution of an aggregated Markov Reward Model where each state represents a cluster. The aggregated model is governed by a transition matrix **B**, the initial probability distribution π_B, and a reward structure ρ_B.

The probability of a transition occurring between states I and J, representing clusters I and J, is given by equation (6.39), where p_{ij} is the probability that a transition occurs from state i to state j.

$$b_{IJ} = \sum_{i=1}^{n(I)} \sum_{j=1}^{n(J)} p_{ij} \tag{6.39}$$

The probability that the aggregated model starts in state I is given by the sum of the probabilities of starting in any state i of cluster I.

$$\pi_{BI} = \sum_{i=1}^{n(i)} \pi_i' \tag{6.40}$$

The reward rate associated with cluster I is the mean accumulated reward by cluster I divided by the specified observation period.

The mean accumulated reward of this aggregated model can then be determined.

6.8.7 Validation against the actual overall network model

When the clusters are connected, the individual performance models need to be merged. If performance is expressed in terms of a value such as number of packets queueing in the network, the performance associated with the whole network is the sum of the performance measures for the different clusters. This expression ignores the number of packets queueing across the inter-cluster links, unless the inter-cluster links are included as part of one of the clusters they connect. The underlying assumption has been that only single link failures may occur so that if a failure occurs in any cluster, the other clusters are judged to be fully operational. Therefore, in the construction of the performance vector for the whole network, given a cluster i, the performance level associated with any state of that cluster is given by the sum of the performance level of all the other clusters under fully operational conditions and the performance level associated with that state during design of the cluster.

$$Perf_{overall}(i, I) = Perf(i, I) + \sum_{\substack{J=1, \\ J \neq I}}^{L} Perf(0, J) \tag{6.41}$$

where

$Perf_{overall}(i, I)$: performance of the overall network in state i of cluster I,

$Perf(i, I)$: performance of cluster I in state i and

$Perf(0, J)$: performance of cluster J in the fully operational state.

In order that the results obtained after aggregating the clusters are comparable to the results obtained for the whole network, it is necessary that the reward associated with any

state of the whole network is equal to the reward associated with that particular state in the cluster where it appears. Therefore, when comparing the aggregated results with the actual results for the overall network, the mapping of performance on to rewards has to be done in such a way that the above requirement holds. This only simplifies the analysis.

6.8.8 Establishment of a stopping criterion for design

Assume that the performability specification given for the whole network is equal to $x_{spec}(t)$. Let the error involved in the aggregation process be bounded by e_{max}, that is

$$|x_{spec}(t) - x_{agg}(t)| < e_{max}, \tag{6.42}$$

where $x_{agg}(t)$ is the performability measure obtained when results for the individual clusters are aggregated.

Therefore, to ensure that performability specifications are met when the clusters are merged, a suitable stopping criterion is given by $(x_{spec}(t) + e_{max})$. The iterations of design may proceed until the aggregated performability is equal to, or exceeds, the criterion, at which stage it is expected that the actual performability of the network will meet the specifications. Another way of viewing this is to stop the design at a stage where $(x_{agg}(t) - e_{max})$ exceeds the specifications $x_{spec}(t)$.

In the case where aggregation is carried out according to equation (6.37), this can be extended to be the stopping criterion for design of each cluster. If each cluster is designed to meet an objective performability measure of $(x_{spec}(t) + e_{max})$, application of equation (6.37) for aggregation should give an overall aggregated performability of $(x_{spec}(t) + e_{max})$.

6.8.9 Error analysis

Under steady-state conditions, the error involved in estimation of the equilibrium probabilities is in ε, where ε is the degree of coupling between the clusters [COU77]. Therefore, as explained before, the maximum error with which the mean accumulated reward can be estimated is $D\varepsilon$, where the number of states in the overall model is D.

In transient analysis, estimation of the error bounds is more difficult to calculate. Error bounds on the accuracy of approximate procedures and truncations are usually not available. Van Dijk [VAN91] proposed conditions which enable the estimation of an error bound when comparing transient reward structures for a continuous-time Markov chain under slightly different parameter values or system protocols. Another route for determining error bounds is through a sensitivity analysis of the system with respect to variations in its parameters. Heidelberger and Goyal [HEI88] derived the sensitivity of the transient solution of a continuous-time Markov chain with respect to a parameter of the transition rate matrix. Grassi and Donatiello [GRA92] evaluated a closed form expression for the derivative of the cumulative reward probability over a finite time interval, where both transition rates and reward rates are assumed to be a function of a given system parameter; this enables a sensitivity analysis in a performability framework.

The proposed approach for an estimation of the error involved in the decomposition and aggregation process applied to the Markov chain when a transient analysis is carried out is as follows. For each cluster, an estimate of the error which arises from the decomposition process may be obtained by considering the system to be such that it consists of the cluster and an additional state which represents the remaining clusters. The transition rates into that aggregated state are the sum of the failure rates of the other components of the network, which do not form part of the cluster being designed. The transitions out of that state take the system into one of the states of failure associated with the cluster. The probability that the system starts in that state at time equal to zero is the sum of the probabilities of starting in the states associated with the other clusters. The reward associated with this state can be given by the aggregated mean accumulated reward normalised by the observation period, where aggregation is carried out according to equation (6.37), that is the mean accumulated reward of each cluster is weighted by the probability of the network starting in a state of that cluster.

Under the assumptions that the remaining components of the network are considered as a single state, this approach can only provide an estimate of the error incurred in considering the cluster on its own. The difficulty arises in relating these individual errors

to the error involved when comparing the aggregated measure of interest and this value for the whole network.

A crude upper bound on the error can be in the form given by equation (6.43).

$$\text{Upper bound of overall error} = \sum_{i=1}^{L} \text{error in cluster } i \qquad (6.43)$$

6.9 Results

In this section, results are presented for the same network of ten nodes, designed previously in Chapter 5, except that it is now designed as two clusters of five nodes each. Physical distance is used as criterion for clustering the nodes. The design has been carried out according to the different scenarios identified in Sections 6.4 and 6.5 and the results for each case will be discussed.

6.9.1 Example 1: design under steady-state conditions

Although a steady-state analysis is not particularly suitable for evaluation of performability, this example is considered because error bounds exist when the steady-state probability distribution is estimated after decomposition and aggregation.

In this example, the clusters are designed and connected to meet performability specifications given in terms of the mean accumulated reward under steady-state conditions. The initial minimal spanning tree connecting the whole network is first derived prior to clustering of the nodes. Once the nodes are grouped, links are added to each cluster configuration so that at least a 2-connectivity criterion holds for each of them. Fault-tolerance augmentation can then be applied to each cluster. The approach taken to design follows the process described in Section 6.7.5. The steady-state probabilities are calculated for each cluster. The transition matrix governing the aggregated system is then constructed, and used to solve for the approximate steady-state probabilities of the network. These values can be used to estimate the reward which can be accumulated by each cluster. Design of the network is based on capacity augmentation solely so that there is no need to re-evaluate the steady-state distribution, which would be required if a link was added to the network.

Therefore, the network which consists of a total of 11 intra-cluster links is represented by a Markov Chain of 13 states, 7 states for cluster 1 and 6 states for cluster 2. The inter-cluster links are assumed to be very reliable so that the possibility that they may fail can be neglected.

Two cases will be considered, and it will be shown that although the conditions for near complete decomposability hold, they do not necessarily translate in a suitable means of establishing a stopping criterion for design. In the first case, the failure rates varied between 1 and 5 per 100 days. After construction of the transition matrix, ε is evaluated to be 0.105. This value of ε which satisfies the criterion for near complete decomposability, given by equation (6.20), translates into a maximum error of 1.365 in the mean accumulated reward. Given that rewards range from 0 to 1, such a value for the error does not make sense.

In the second case, the failure rates have been reduced by five times, so that ε is now equal to 0.021 and, therefore, the maximum error expected in the mean is 0.273, which is a more reasonable value. Let the specifications be set at 0.98 for the whole network. Given that the steady-state probability associated with cluster 1 is 0.6406 and hence 0.3594 for cluster 2, the specifications can be viewed as such:

Specification for cluster 1 = (0.98 x 0.6406) = 0.6278
Specification for cluster 2 = (0.98 x 0.3594) = 0.3522

Maximum error in accumulated reward for cluster 1 = (0.021 x 7) = 0.147
Maximum error in accumulated reward for cluster 2 = (0.021 x 6) = 0.126

Therefore, the maximum accumulated reward aimed at to ensure that specifications are met should be as follows:
Maximum accumulated reward aimed at for cluster 1 = (0.6278 + 0.147) = 0.7748
Maximum accumulated reward aimed at for cluster 2 = (0.3522 + 0.126) = 0.4782

Since both these values exceed the maximum achievable accumulated reward for each cluster, the maximum is then set to be 0.6406 for cluster 1 and 0.3594 for cluster 2.

Results show that at a stage where the estimated mean of the network is 0.9921, the actual mean is equal to 0.9862, at which point the specifications are met. The error between the estimated and actual means is 0.0059. The value ε provides a means of establishing the error involved, and enables fixing a maximum bound on the error, thereby making it possible to aim at a certain maximum specification. However, this can lead to overestimation of the degree of augmentation required by the network, as shown in the example. Overall specifications were satisfied even before reaching the maximum aimed specification for each cluster.

6.9.2 Example 2 - design under transient conditions

In this case, design proceeds in the same manner as in example 1 in the sense that inter-cluster links are established prior to design of the cluster. However, a transient analysis is carried out, where the network is assessed over a period of ten days. The way in which the aggregated measures of performability compare with the actual measure for the whole network is investigated.

The failure rates of the links constituting the network range between 1 and 5 per 100 days whereas the average repair time varies between 22 and 24 hours so that ε is equal to 0.105, and satisfies the sufficient criterion for near complete decomposability.

In this case, both the addition of links and capacity augmentation are used to equip the network with more resources. The results obtained for the network, in terms of mean accumulated reward, are as shown in Table 6.1. The aggregated means obtained at iteration x use the means of each cluster obtained at iteration x. Aggregated mean 1 corresponds to the value obtained when aggregation is carried out for the completely decomposable system according to equation (6.37). Aggregated mean 2 corresponds to the value obtained when the process is carried out as explained in Section 6.8.6.2. From the table, the aggregated means are overestimates of the actual mean accumulated reward by the network.

Iteration number	Mean acc. reward of cluster 1	Mean acc. reward of cluster 2	Aggregated Mean acc. reward 1	Aggregated Mean acc. reward 2	Actual Mean acc. reward
0	8.7004	9.0675	8.8840	8.8242	8.0039
1	9.3594	9.2213	9.2904	9.3114	8.7632
2	9.3594	9.3982	9.3788	9.3708	8.9255
3	9.3818	9.6163	9.4991	9.4529	9.1354

Table 6.1 - Comparison of means of the clusters, aggregated, and whole network

In Section 6.8.9, a method was proposed whereby the error involved in design of each cluster, while ignoring the existence of the other clusters, could be estimated. The remaining components of the network were aggregated into one state, which is included in the cluster model. In Tables 6.2 and 6.3, this error is evaluated. At iteration x, the mean accumulated reward of cluster 1 is computed for the cluster on its own, as well as for a system consisting of the elements of cluster 1 and one state representing cluster 2. The reward rate attached to the state is the mean accumulated reward associated with cluster 2 normalised by the observation period, at that iteration. The converse process is applied to determine the error for cluster 2.

Iteration number	Mean acc. reward of cluster 1	Mean acc. reward of cluster 1 and one aggregated state	Error in the mean acc. reward
0	8.7004	8.3607	0.3397
1	9.3594	9.1099	0.2495
2	9.3594	9.1405	0.2189
3	9.3818	9.2571	0.1247

Table 6.2 - Table of results for cluster 1

Iteration number	Mean acc. reward of cluster 2	Mean acc. reward of cluster 2 and one aggregated state	Error in the mean acc. reward
0	9.0675	8.3451	0.7224
1	9.2213	8.8429	0.3784
2	9.3982	9.0840	0.3142
3	9.6163	9.2756	0.3407

Table 6.3 - Table of results for cluster 2

It was proposed that the sum of the individual errors estimated for the clusters could provide an upper bound to the error involved in the aggregation process. A comparison of this sum of errors and the actual errors involved in the aggregated means is given in Table 6.4. As can be seen, the sum of errors is much higher than the actual calculated errors, and does not offer a tight bound.

Iteration number	Sum of errors in mean acc. reward of clusters	Error in aggregated mean acc. reward 1	Error in aggregated mean acc. reward 2
0	1.0621	0.8801	0.7821
1	0.6279	0.5272	0.5685
2	0.5331	0.4530	0.4417
3	0.4654	0.3637	0.2895

Table 6.4 - Comparison of errors involved in aggregation

From these results, it cannot be said that any of the aggregation mechanisms is better because, although for most iterations, aggregation according to the second approach yields a better result, in that it is closer to the actual mean, at the first iteration, the reverse occurs.

If the sum of errors is used as a bound on the error incurred in aggregation, it can be applied in the following way. The aggregated mean x_{agg} is computed, and if the value obtained by (x_{agg} - error) exceeds the specifications, design is stopped. It is expected that the

actual performability measure of the network would meet specifications at that stage. To give an example of this, consider Table 6.5 where the values (x_{agg} - error) are compared with the actual mean.

Iteration number	Aggregated Mean 1 - Sum of errors	Aggregated Mean 2 - Sum of errors	Actual Mean acc. reward
0	7.8219	7.7621	8.0039
1	8.6625	8.6835	8.7632
2	8.8457	8.8377	8.9255
3	9.0337	8.9875	9.1354

Table 6.5 - Comparison of lower bounds on expected means and actual mean

Suppose the specifications were set at 8.8. In this case, at iteration 2, the difference between both the aggregated means and the sum of errors exceeds the specifications, indicating that the actual mean has to meet the requirements, which is true. If the specifications require the mean to be 8.9, an additional iteration is needed before it can be established that the network is sufficiently equipped to give this mean, although the overall specifications were satisfied at iteration 2.

Although the sum of errors gives a crude bound on the error involved in aggregation, the relation between these individual errors and the actual error has not been established so that the tightness of the bound is not known. More investigation into this aspect is required.

6.9.3 Example 3: *worst-case* scenario

In the remaining examples, the different scenarios derived in Section 6.5, where inter-cluster connections are made after design of the clusters, are investigated. In the *worst-case* scenario, the approach is to design the most robust network assuming that only a pair of unidirectional links connects two clusters. Inter-cluster traffic is assumed to arrive at, and leave the network via a single internal node. Once the nodes have been grouped, the minimal spanning tree for each cluster is derived.

In design of a particular cluster, the following approach is taken. The link capacities are evaluated for each case when each internal node is connected to the virtual external node. Each link is assigned the maximum capacity obtained out of the capacities calculated for these different possibilities. The topology, with this set of link capacities, associated with the worst performability when comparing the different possibilities of connections between internal nodes and the virtual external node, is then chosen for augmentation.

This procedure is applied to the network of ten nodes, designed in two clusters of five nodes each. Each cluster is designed until all states are associated with a reward of 1.0. Inter-cluster connection is established by links added between the internal nodes previously connected to the virtual external nodes. Evaluation of the network when the clusters are linked shows that in the presence of only one pair of inter-cluster links, these requirements are satisfied, that is, each configuration of the network is performing at a level corresponding to a reward of 1.0. When another pair of inter-cluster links is added, the performance in all states except those where the inter-cluster links fail, is improved significantly.

This network has been designed to be as robust as possible to face the conditions when there is only one means of communication between the clusters. As a result, it is a high-capacity network, which inevitably performs very well when intra-cluster links fail, or when the network is failure-free.

6.9.4 Example 4: *best-case* 1 scenario

Designing the clusters according to this scenario assumes that each internal node is connected to a virtual external node via which it communicates with external clusters. Therefore, the handling of external traffic, that is, inter-cluster traffic puts minimal load on design of each cluster. The procedure again consists of deriving the minimal spanning tree that connects the nodes, adding links to satisfy a 2-connectivity criterion, and then augmenting the network to meet performability specifications. The proposed approach for inter-connecting clusters is to establish connections between the nodes where there is most demand for traffic.

6.9.5 Example 5: *best-case n* scenario

In this scenario, each internal node is assumed to be connected to each and every external node with which it needs to communicate. This leads to easy establishment of inter-cluster connections. The design approach is similar to the *best-case* 1 scenario, the only difference lying in the way in which external traffic is handled.

6.9.6 Comparison between the scenarios

A comparison of the networks designed according to the different scenarios is carried out for cluster 1. Results show that with respect to internal traffic only, the performance levels delivered by the different configurations are comparable although the capacity of the internal links constituting the cluster designed according to the *worst-case* scenario is almost three times the capacity of these links when design proceeds according to the two other scenarios. The reason is because the extra capacity allocated when the *worst-case* scenario is used is assigned for external traffic. However, when traffic across the external links is also taken into consideration, the *worst-case* scenario is, as expected, associated with the best performance levels, followed by the *best case* 1 scenario, and lastly the *best case n* scenario. Again, the cluster designed according to the *worst-case* scenario is highly equipped in terms of capacity as compared with the other two, which are almost equivalent. The difference in performance between the topologies generated by the two best-case scenarios is due to the fact that the performance across a single high-capacity link tends to be better than the performance delivered if the traffic is distributed among a number of lower capacity links. This is the reason why the *best-case* 1 scenario performs better although from a reliability point of view, the *best-case* n scenario is preferable.

Table 6.6 summarises results obtained for cluster 1, in terms of the capacity of the cluster, whereas in Table 6.7, a comparison of the performance of the cluster in different states is given. The performance in the state of failure corresponds to the maximum performance level attained in all the possible states of failure of intra-cluster links. Performance is being measured in terms of the average number of packets queueing in the network and includes packets across the inter-cluster as well as the intra-cluster links.

Capacity (data units/s)	Worst-case	Best-case 1	Best-case n
Capacity of initial cluster including external links	792094	345902	347251
Capacity of initial cluster excluding external links	641868	195409	195646
Capacity of final cluster including external links	1128094	489902	491251
Capacity of final cluster excluding external links	977868	339409	339646

Table 6.6 - Comparison of cluster 1 designed according to the three scenarios in terms of capacity

Order of performance (no. of data units) in	Worst-case	Best-case 1	Best-case n
fully operational state, including external traffic	510	2100	3900
fully operational state, excluding external traffic	4	5	5
state of link failure, including external traffic	520	2500	4000
state of link failure, excluding external traffic	6	17	17

Table 6.7 - Comparison of the performance of cluster 1 designed according to the three scenarios

After connecting the clusters, the performance of the overall network can be evaluated in all the states in which it can be. Table 6.8 gives an indication of the order of performance levels attained when intra-cluster and inter-cluster links fail. In the network designed according to the *worst-case* scenario, two pairs of links connect the clusters so that if a pair fails, no disconnection occurs. As can be seen, the high capacity of the network is reflected in the good performance of the network as compared with the networks designed according to the best-case scenarios. Two pairs of inter-cluster links contribute to a significant improvement in performance as compared with the performance of cluster 1.

As already explained, in the best-case scenarios, the difference in the performance levels in the fully operational state and a state where an intra-cluster link fails, results from directing traffic across a number of small-capacity inter-cluster links in the *best-case n* scenario instead of a single high-capacity link in the *best-case* 1 approach. However, failure

of an inter-cluster link in the *best-case* 1 scenario causes more deterioration in performance due to loss of more capacity. The performance of the network is better than that of the cluster because of the following reason. In the best-case scenarios, the capacity of the inter-cluster links has been chosen to be the maximum of the values obtained for these links in design of the two clusters individually. For example, if the capacity of the link carrying outgoing traffic from an internal node *a* in cluster 1 to some external node *b* in cluster 2 is worked out to be *x*, and the capacity of the link carrying incoming traffic from node *a* to node *b* is calculated to be *y*, the higher value out of *x* and *y* is chosen for the capacity of the link carrying traffic from node *a* to *b*, when they are connected. This causes the performance of the network to be better especially when the *best-case n* scenario is applied.

Order of performance (no. of data units)	**Worst-case**	**Best-case 1**	**Best-case n**
in the fully operational state	13	207	3858
when an intra-cluster link fails	20	250	4000
when an inter-cluster link fails	500	400000	200000

Table 6.8 - Comparison of the performance of the networks designed according to the different scenarios

6.10 Conclusions

In this chapter, the main objective has been to investigate a clustered-based approach to a design technique based on performability. Given the performability-based approach, design in clusters requires a decomposition of both the reliability and performance models of the network.

In decomposition of the performance model, the question that arises is to how to predict the performance of the overall network when the clusters are connected, given the performance of the individual clusters. Although it is clear that if performance is measured in terms of a value such as network delay across each link, then the performance of the network is the sum of the performances of the different clusters, this is only the case if the inter-cluster connections do not cause a redistribution of traffic. This question of how to handle inter-cluster traffic during design of each particular cluster has led to identification

of a number of possible scenarios. In the *worst-case* scenario, during design of a particular cluster, each external cluster is treated as a single virtual external node. In the *best-case n* scenario, the nodes of the cluster being designed are assumed to be directly connected to all the external nodes with which they need to communicate. The *best-case* 1 scenario falls in between the two in that each internal node is assumed to be connected to a virtual external node representing the set of nodes with which communication is required.

In order to reduce the effects of the problems arising from handling inter-cluster traffic, it was assumed that only one pair of inter-cluster links exist in the investigation of how the techniques of decomposition and aggregation could be applied to the network model. As explained, the transition matrix associated with the overall network is not in a decomposable form; a transformation was proposed to generate a transition matrix which characterised the same behaviour but was in a nearly completely decomposable form. The new transition matrix with its associated state probability distribution and reward structure represent exactly the same system as the original model. Application of this transformation makes it possible to relate the clusters' models to the overall network model and to decide on suitable aggregation mechanisms for the network.

Once the clusters are designed and evaluated, an aggregation of the evaluation measures of the clusters is required to give an estimate of this measure for the overall network. Two aggregation techniques have been considered, whereby the first approach yields exact results for the fully completely decomposable system; in other words, the interactions between the subsystems are assumed to be equal to zero. In the second approach, each cluster is represented by a single state to construct an aggregated model of the network, which consists of as many states as there are clusters. The reward associated with each state corresponds to the average accumulated reward by that cluster normalised by the specified observation period. The average accumulated reward for this system is then evaluated to give an estimate of this value for the nearly completely decomposable system.

The question that requires more investigation is how to bound the error involved in this decomposition and aggregation process on the average accumulated reward. If a suitable bound can be found, a stopping criterion for the aggregated model, or for the

individual clusters, can be derived. In transient analysis, establishment of such a bound is difficult and generally, expressions involving bounds on errors when approximate procedures are used are not available. A crude error bound has been proposed, which consists of the sum of the errors involved in the design of each cluster on its own, where this error is calculated after considering the cluster model as consisting of states for its components and an additional state where the remaining components of the network have been aggregated. A network example has shown how this bound can be used, but the tightness of the bound has not been found. This requires establishment of the relation between the errors occurring in each cluster model and the overall error resulting from aggregation. The error involved in computing the steady-state distribution associated with the overall network is related to the degree of coupling between the aggregates, that is, the clusters. However, examples have shown that use of this error bound does not always translate into a suitable stopping criterion in design of clusters. It can also lead to overestimation of the stopping criterion, resulting in more resources being put into the network than are necessary.

Examples have also been given when the network is designed according to the various scenarios which arise from different considerations of inter-cluster traffic. In the *worst-case* scenario, the network is designed to face a condition where only one pair of inter-cluster links exist so that it is very robust and highly equipped in terms of capacity; as a result, it is also expensive and resources do get wasted when inter-cluster links are operational. In the two other scenarios, referred to as the best case scenarios, inter-cluster traffic places minimal load on design of each cluster; consequently, the clusters are adequately equipped to cope with failure of an intra-cluster link but more augmentation is required for the overall network to be able to face a state where the inter-cluster links fail. These networks are however cheaper.

The examples considered in this chapter have included only two clusters. The case where inter-cluster links are established before design of each cluster can be easily extended to design of multiple clusters. The decomposition and aggregation stages proceed according to the same steps used in the design of the two-cluster network. In the *worst-case* scenario, in design of any particular cluster, all other external clusters may each be represented by a

virtual external node. In the *best-case n* scenario, as the number of clusters increases, the fully connected inter-cluster topology increases significantly in size and intricacy. In the *best-case* 1 scenario, each internal node may be assumed to be connected to a number of virtual external nodes, where each external node represents the nodes of a particular cluster.

The general conclusion from a clustered-based approach to performability-based design is that such a method would be viable if the error involved in aggregation and decomposition could be closely bounded. If the error bounds are not tight enough, more augmentation might be put into the network than is required to meet the specifications. Furthermore, one of the assumptions when investigating how to apply decomposition and aggregation to the network model has been that a single pair of inter-cluster links exist between clusters, which are judged to be very reliable. In practice, more inter-cluster links are required. These contribute to a different distribution of traffic across the network, leading to an even better performance. Therefore, to make sure that only enough resources are put into the clusters to meet the specifications when more inter-cluster links are added, design of the clusters needs to be stopped after a number of iterations, and the performance of the network in its various states can be evaluated after linking the clusters. Another requirement for such an approach to be good is that the failure rates of the components be fairly low, in order that the degree of coupling between the clusters is small. Under conditions that only single link failures may occur, the degree of coupling is a sum of the failure rates of the components.

From a computational point of view, a clustered-based approach is advantageous because in design of each cluster, fewer states need to be considered, leading to faster evaluation of the performability measure required. Another factor which contributes to quicker evaluation is that in consideration of each cluster, the maximum rate Λ is less than the value obtained if the whole network model is used. Recall that Λ is the maximum of the sum of rates out of each state of the model. Therefore, since Λ is used in determination of the number of transitions required for computation of performability to meet the pre-specified error tolerance, the number of transitions needed is less. A lower number of transitions and states in the model contribute directly to a reduction in the memory requirements. Given the gain in computation time and memory requirements, if

multiple failures or partial repair were to be considered, such an approach would be more appropriate.

Although the analysis of a network in terms of clusters is interesting in selecting appropriate means of decomposing the system, and aggregating measures of interest, the synthesis of the network is a more difficult problem to consider. After this initial investigation of performability-based network design in clusters, the desirable approach is to cluster the nodes such that inter-cluster traffic is minimal with respect to intra-cluster traffic and thus has less effects on performance as inter-cluster links are established. Secondly, the error involved in the decomposition and aggregation process needs to be tightly bound in order to be able to translate this into a good stopping criterion for design in clusters.

Chapter 7

Conclusions and the way ahead

7.1 Conclusions

The main objectives of the thesis have centred around an investigation of the use of performability in the design of communication networks. Performability is a concept which originated in the early 1980's due to the shortcomings of conventional separate performance and reliability analyses. Performance and reliability have usually been investigated separately although they are closely related. On the one hand, performance analysis precludes the occurrence of any failures, and assumes that all components of the system are operational. Reliability evaluation on the other hand views the system as being binary-valued with respect to performance, that is, it is either operational or has failed, irrespective of the range of performance levels exhibited. Individual assessments of performance and reliability are difficult to combine to reflect the operation of a system over a given observation period, when its performance varies across a range as a result of the occurrence of failure and repair actions. This is especially the case for fault-tolerant systems where a component failure may reduce the quality of service delivered without necessarily entailing complete system failure.

The concept of *performability*, a combined measure of performance and reliability, was introduced as a means of evaluating these types of systems. Performability evaluation is based on observing the variations in the performance delivered by a system as it evolves through different states in the presence of failures and subsequent repair actions. The performability of a system quantifies, in terms of probability, its ability to perform in an environment where it is affected by failure and repair actions. Performability evaluation involves construction of a suitable model and its solution. The Markov Reward Model is particularly appropriate as means of modelling because while the underlying Markov Process represents the evolution of the system through different configurations due to failures and repairs, its associated reward structure gives an indication of performance in

each state. Therefore, the model enables a combination of both performance and reliability data. A number of techniques exist for the solution of the model. The method chosen in this thesis is based on the *randomisation technique* [DES89]. A treatment of the definition, model construction and solution of performability constituted the core of Chapter 3.

In Chapter 3, the measure of performability sought is a distribution of average accumulated reward, which given a mapping of performance on to reward should give an indication of a distribution of average performance. An investigation of the effects of varying performance and reliability data on this measure led to the statement that a distribution of average accumulated reward can be used as a measure of the fault-tolerance of the system. In view of these results, the concept of a performability index was introduced as a means of assessing the relative importance of the constituent components of a system by virtue of their rate of failing, their average repair times, and the degree of performance degradation caused by their failure. Two definitions of the index were proposed and an example was used to demonstrate how the index can be used in assisting where to incorporate redundancy in a system.

During application of the randomisation technique, a number of problems were encountered. One of the problems was associated with numerical errors introduced due to manipulation of very large numbers. This contributed to the distribution of average accumulated reward being non-monotonic. A suitable check was devised and implemented to avoid these types of errors. Another problem arose because of the many-to-one mapping of performance on to reward, with the result that it is possible that a state of failure is associated with the same reward as the fully operational state so that the degradation occurring in the presence of that failure is not reflected. In such cases, increasing the rate of failing, or increasing the average repair time, may lead to the system appearing to be more fault-tolerant, which is in contradiction with expected results. Hence, in the comparison of different systems, a necessary condition is that the reward associated with a state of failure is not the same as that attached to the fully operational state in order to reflect the degradation caused. Evaluation of performability is affected computationally by the size of the state space. A number of techniques exist in the literature, which attempt to overcome the limitations imposed by the number of states constituting the model. In this thesis, the

lumping technique has been chosen as means of reducing the number of states in the model whenever possible.

Given a technique to evaluate performability, and more importantly given a means of assessing the fault-tolerance of a system, its application in the design of communication networks can be investigated with a view to generating fault-tolerant networks. This constitutes the main objective of the research carried out.

The process of optimal design is very complex due to the large number of variables and constraints that need to be taken into consideration. In an attempt to simplify the problem, Gerla and Kleinrock have decomposed it into three sub-problems, each of which can be solved separately. These sub-problems are topological design, capacity assignment, and routing. A complete review of the design process was presented in Chapter 2, with particular attention being given to algorithms for topological design and capacity assignment. Existing techniques of fault-tolerant design based on capacity augmentation were also considered. A study of design techniques and their relative merits and shortcomings with respect to performance and reliability analyses led to proposal of the novel performability-based design method in Chapter 4.

In accordance with Gerla and Kleinrock's decomposition, the proposed design technique consists of three stages where the sub-problems of topological design and capacity assignment are solved while assuming optimal routing. In the first two stages, a network which is minimally 2-connected is derived; basic reliability requirements are therefore satisfied. The third stage, the *fault tolerance augmentation* stage, which is the most important stage, is then required to augment the network sufficiently to meet performability requirements. The augmentation mechanisms are adding new links and increasing the capacity of existing links. Decision as to where addition of a new link or a capacity augmentation is more beneficial is based on performability.

A typical iteration of the fault-tolerance augmentation stage proceeds as follows. At the beginning of an iteration, the performance of the network can be determined in all its possible configurations. A state has to be chosen for investigation of where to apply

augmentation; the chosen state is a state of failure, which can be either the state of worst performance, or the one associated with the worst performability index. Once the state has been selected, the load on the links needs to be determined. A set of rules have been derived where, depending on the distribution of traffic among the links, different possibilities regarding where to add potential links, or apply capacity augmentation, are identified. Each of these mechanisms is tried, giving a topology whose cost and performability can be established. The topology which achieves the best improvement in performability at a minimal increase in cost is then selected. The main aspects of this design strategy that differ from conventional design are that firstly, optimisation is applied in a state of failure instead of the failure-free state, and secondly, choice of the best topology is based on performability instead of performance in the fully operational state.

During development of the design algorithm, a number of issues have to be taken into consideration, namely definition of the cost of the network, choice of a suitable performance measure, and choice of the routing algorithm. Two definitions have been proposed for cost of the network; in both cases, the cost of a network is expressed in terms of the cost of its components, but the second definition also includes the average repair cost expected to be incurred over the observation period. Given that the system's performance is being appraised over a finite period of time, it is only reasonable to consider the average cost of maintaining the network besides the cost of setting it up.

An investigation into the choice of a suitable performance variable led to the proposal of a new technique to evaluate a network which can be subjected to link as well as node failures. A node failure can cause the resulting configuration to be associated with a better performance if the amount of traffic lost due to the failure is not taken into account. Traffic destined for the node may be discarded, and traffic originating from that node is no longer generated. Therefore, from the performance data collected, it appears that a node failure is more beneficial. In order to obviate such assessments, the approach adopted is to consider both the performance measure, for example delay, and the amount of traffic lost when evaluating a network. A maximum delay is imposed so that any packets exceeding this delay are discarded. This entails loss of traffic even under fully operational conditions,

but provides a more appropriate means of judging the degree of degradation caused by component failures.

Choice of the routing algorithm is another area which was investigated. Different routing algorithms have been tried and it was observed that in a particular example, the configuration resulting from a link failure was associated with a better performance. The reason such a feature was exhibited was due to the non-optimality of the routing algorithm which did not attempt to optimise performance. In view of these results, it is desirable to use a routing algorithm that attempts to optimally distribute traffic in order to achieve the best performance in a particular configuration. It does not make sense to try and augment a network in order to achieve a certain level of performance and performability if the same effects can be obtained by changing the routing strategy at the very outset. Therefore, an algorithm based on the Gradient Projection Method was chosen for routing purposes in the network design.

Chapter 5 was devoted to an appraisal of the capability of the proposed design technique, its relative merits and shortcomings *vis-à-vis* conventional design approaches. Results show that the proposed design approach yields more fault-tolerant networks than traditional design, and may even do so using less resources. In other words, the performance, over a period of time, of the network designed according to the new method is better than the performance delivered by a network designed according to a traditional scheme. It is possible that the latter network performs better under failure-free conditions, but the former network performs at an acceptable level in that state as well as in its states of failure. Therefore, in the presence of failures, the performance of the network does not degrade drastically. As previously mentioned, the reason behind this improvement is that firstly, the proposed design technique optimises in a state of failure, and secondly chooses the network that globally performs best on average at the end of each iteration. The attraction of this performability-based approach is that in some cases, due to a better placement of resources, a more fault-tolerant network may be obtained using less links or capacity, which has a direct bearing on the cost of the network.

Further investigation into choice of the augmentation mechanism showed that the mechanism which is most beneficial depends on the situation being dealt with as well as the traffic distribution across the network. Although, it appears that the routing algorithm would tend to favour a capacity augmentation to addition of more links, such a statement is not necessarily true. Both mechanisms have to be tried depending on the scenario which arises. The design algorithm was also applied to a case where partial link failures are allowed. Results are no different in the sense that the proposed technique outperforms traditional design in its generation of more fault-tolerant networks.

The proposed design strategy is however plagued by the fact that it is computationally more intensive. The performance of the network has to be evaluated in all the possible configurations in which it can be prior to performability evaluation where computational and memory requirements increase drastically with an increase in the size of the state space. In an attempt to circumvent these problems, a performability-based design technique, based on the concept of clustering was proposed and investigated in Chapter 6. A number of issues have to be taken into consideration both at the design and the modelling stages. From a design point of view, questions arise as to how inter-cluster traffic is handled during design of any particular cluster. This led to consideration of different types of scenarios with their relative advantages and disadvantages. From a modelling viewpoint, the problem amounts to decomposing the model, evaluating each sub-model, and then aggregating the results to give an estimate of the overall network being designed. Application of the well-founded theory of decomposition and aggregation to network design was investigated. Performability-based design of networks in clusters requires a decomposition of the reliability as well as the performance models. The transition matrix characterising the network was not in a decomposable form. A transformation which involved expanding the network model was proposed in order to generate a transition matrix which is in a nearly completely decomposable form. This new model represents exactly the same system behaviour as the original model. This transformation facilitated relating the clusters' models to the overall network's model. Two means of aggregation were considered to yield estimates of results for the whole network using measures obtained for the clusters, when a transient analysis is carried out. A crude bound on the overall error involved in the aggregation process was expressed in terms of the error involved in design

of each cluster on its own. However, the tightness of the bound remains to be established, which would make performability-based design of networks in clusters viable. The advantages of this approach dwell in the gain in computational time and memory requirements in design of each cluster.

7.2 The way ahead

Further research and investigation into the following aspects is worth pursuing.

1. The proposed definitions for the performability index are applicable to a single failure/ total repair scenario. The concept needs to be extended to a more general framework where multiple failures and partial repair can occur. In this way, it is possible to identify a group of components that is most significant in a system instead of a single component. The effects of using different measures of performability as the performability index need to be determined. This can lead to choice of a more appropriate definition of the index.

2. Recently, Nabli and Sericola [NAB96] have proposed a new algorithm to compute the performability distribution where the number of operations is linear in respect of the number of states and linear in respect of the number of rewards. This implies that the computational problems arising from the size of the state-space and the number of rewards considered are reduced. If more rewards are used, the mapping of average accumulated reward on to average performance becomes more accurate. The effects of using this algorithm in the proposed design method need to be investigated.

3. The proposed design approach has been studied under the assumptions that only single link failures may occur. The capability of the technique needs to be investigated to the full in a more realistic setting where multiple link failures, requiring different global or local repair actions, can occur. Consideration of the states resulting from the occurrence of these structural changes may lead to different augmentation mechanisms being applied at each iteration of design. However, it also contributes to an increase in the size of the state-space.

4. The design technique can be extended to cope with node failures. In Chapter 4, a suitable performance measure to evaluate a network where node failures can occur was identified. The use of this performance measure in the design process can be investigated. Furthermore, different augmentation strategies and rules need to be derived as to how to equip the network in order to minimise the degradation caused by a node failure. When a node fails, the degradation in performance is considerably more than that resulting from a link failure. However, usually the possibility of a node failing is rare as compared with the probability that a link fails. Therefore, the use of an index which gives an indication of the significance of each component in the network is even more important in order to aid in the choice of the state where augmentation is to be applied. This issue requires further investigation.

5. The application of sensitivity analysis to design is worth studying. This would enable assessing the effects of varying design parameters related to reliability as well as performance data on the performability of the network.

6. Bounds on the error involved in the decomposition and aggregation process when a transient performability analysis is carried out need to be found and the tightness of these bounds has to be established. The use of these bounds in the design of networks in clusters can then be investigated.

7. In this thesis, the design technique has been derived for fairly general networks where only one type of traffic is considered. Choice of the performance measure which is used depends on the traffic being carried. The way in which performability can be used in evaluation of ATM-based networks where different types of traffic are dealt with can be investigated.

Appendices

In the following appendices, the performance levels of the networks designed are tabulated. For each iteration considered, these values are given for each configuration in which the network can be. The appendices are classified according to the description given below.

Appendix	**Description of set of data**
A1 to A4	Networks are designed using increase in connectivity only
B1 to B4	Networks are designed using capacity augmentation only
C1 to C3	Networks are designed to meet different traffic demands
D1 to D3	Intermediate states of failure are also included

The entries in the **conf** (configuration) column represent the configuration in which the network's performance is obtained and are used as follows.

Entry	**State**
Fully op	Fully operational state
i-j	State where links between nodes *i* and *j* fail
i-j half	State where links between nodes *i* and *j* operate at half-capacity

TOP*x* indicates the topology obtained after *x* iterations of the design approach. Each entry **TOP*x*** is given with the augmentation mechanism used. The notation used is explained in the following table.

Notation	**Mechanism**
add (*i-j*)	Add two unidirectional links between nodes *i* and *j*
aug (*i-j*)	Augment the capacity of both links connecting nodes *i* and *j*
aug *i-j*	Augment the capacity of the unidirectional link from node *i* to node *j*

Appendix A1

Design approach : pure performance approach

Augmentation mechanism : increase in connectivity

Conf.	TOP1 add (3-6)	TOP2 add (3-7)	TOP3 add (2-3)	TOP4 add (3-10)
Fullyop.	49	44	38	36
1-2	1178478	1178470	64	63
1-3	1.085e7	1.085e7	1780830	1780826
2-6	923	914	47	43
3-4	1477047	1477043	1477037	1477035
3-5	2866387	476577	476606	1691
4-5	468	461	456	454
5-6	56	49	42	39
5-8	7230245	1862172	1862171	256173
6-7	136	79	57	49
7-10	122799	114	86	70
8-9	4217697	662627	662609	25292
9-10	3026475	245023	244436	128
3-6	57	50	42	40
3-7		49	44	40
2-3			44	42
3-10				38
P(0.8)	0.4764	0.4408	0.2635	0.1961
Cost	20406.7	21468.7	22210.6	23510.9

Conf.	TOP5 add (3-9)	TOP6 add (3-8)	TOP7 add (4-7)	TOP8 add (1-10)
Fullyop.	36	36	32	30
1-2	63	62	59	58
1-3	1780824	1780823	1780821	103333
2-6	43	42	38	34
3-4	1477034	1477034	79	76
3-5	87	57	49	42
4-5	453	453	34	32
5-6	38	38	35	32
5-8	235	64	48	39
6-7	48	47	43	40
7-10	67	66	47	44
8-9	68	68	47	39
9-10	128	127	59	46
3-6	39	39	35	32
3-7	38	38	34	31
2-3	41	41	39	34
3-10	37	37	33	30
3-9	36	36	32	30
3-8		36	32	30
4-7			36	33
1-10				32
P(0.8)	0.0976	0.0878	0.0216	0.0210
Cost	24736.3	25666.1	26128.1	27575.8

Appendix A2

Design approach : a) performance-based state selection and

b) performability-based topology selection

Augmentation mechanism : increase in connectivity

Conf.	TOP1 add (4-6)	TOP2 add (2-10)	TOP3 add (2-7)	TOP4 add (1-10)	TOP5 add (1-7)	TOP6 add (3-10)
Fully op.	49	38	35	32	30	29
1-2	1178484	74	50	44	39	38
1-3	1.084e7	1780850	116631	95	72	58
2-6	921	45	38	34	32	31
3-4	106	86	82	80	78	78
3-5	3726138	476024	33585	92	64	51
4-5	55	41	38	35	33	32
5-6	60	45	41	37	35	34
5-8	7230238	1723364	415395	275	85	54
6-7	140	56	46	41	40	38
7-10	122964	69218	102	94	54	52
8-9	4217491	857790	246391	220	92	49
9-10	3026484	755312	566053	27765	7768	77
4-6	57	44	41	37	35	33
2-10		49	39	35	33	32
2-7			38	34	32	31
1-10				35	32	31
1-7					32	30
3-10						30
P(0.8)	0.3534	0.1821	0.1614	0.0648	0.0318	0.0
Cost	20739.6	22017.3	22788.9	24731.2	25514.1	26814.4

Appendix A3

Design approach: a) performability-based state selection and

b) performability-based topology selection

Augmentation mechanism : increase in connectivity

Conf.	TOP1 add(2-5)	TOP2 add(5-7)	TOP3 add(5-10)	TOP4 add(5-9)	TOP5 add(4-7)	TOP6 add(1-5)	TOP7 add(1-6)
Fully op.	48	42	40	40	35	34	32
1-2	86	79	80	80	74	74	59
1-3	1783170	1783157	1783149	1783149	1783148	103355	149
2-6	71	55	52	51	44	41	39
3-4	1477045	1477039	1477038	1477037	85	84	82
3-5	2744107	2744107	2743223	2743191	477334	1699	85
4-5	466	461	459	459	38	37	35
5-6	71	55	51	50	43	42	35
5-8	1.262e7	3031837	383043	273	77	76	71
6-7	145	93	87	85	79	76	44
7-10	1457467	43777	43766	43760	65	64	48
8-9	8582203	1561225	111849	104	55	54	46
9-10	7447911	1384099	112816	112814	105	105	57
2-5	57	52	50	50	45	41	39
5-7		48	44	43	37	36	34
5-10			42	41	36	35	33
5-9				40	35	34	33
4-7					40	39	37
1-5						35	57
1-6							34
P(0.8)	0.3021	0.2901	0.2790	0.1961	0.0314	0.0304	0
Cost	20905.7	21974.6	23012.7	23817.3	25264.9	26400.2	27344.2

Appendix A4

Design approach : a) performance-based state selection and

b) performance-based topology selection.

Augmentation mechanism : increase in connectivity

Conf.	TOP1 add (4-6)	TOP2 add (1-6)	TOP3 add (6-8)	TOP4 add (6-9)	TOP5 add (6-3)
Fully op.	49	43	39	37	36
1-2	1178484	1178456	1178446	1178442	1178439
1-3	1.084e7	1958137	1767704	1767583	1767565
2-6	921	462	459	457	455
3-4	106	91	86	82	81
3-5	3726138	1161298	505995	473562	1735
4-5	55	47	43	40	39
5-6	60	50	42	39	37
5-8	7230238	5506119	1849015	543469	255826
6-7	140	125	67	50	49
7-10	122964	25754	25514	25496	25492
8-9	4217491	3005395	2899424	636733	636689
9-10	3026484	1741739	1627350	1627291	1627281
4-6	57	49	43	42	39
1-6		49	44	42	40
6-8			43	39	37
6-9				39	38
6-3					37
P(0.8)	0.3534	0.3394	0.3264	0.3142	0.3028
Cost	20739.63	21683.56	22361.67	23086.76	23723.02

Conf.	TOP6 add(6-10)	TOP7 add (2-5)	TOP8 add (7-8)	TOP9 add (1-5)	TOP10 add (2-10)
Fully op.	35	31	30	30	27
1-2	1178438	55	55	54	30
1-3	1767560	103335	103334	63	39
2-6	454	35	35	34	28
3-4	80	75	75	75	72
3-5	1673	81	81	53	43
4-5	37	33	32	32	29
5-6	35	32	31	31	28
5-8	384	381	377	377	82
6-7	42	38	36	35	30
7-10	25487	25483	36	36	32
8-9	25287	25284	66	66	44
9-10	187269	187265	83	83	43
4-6	38	34	34	33	30
1-6	38	33	32	31	28
6-8	35	31	31	30	27
6-9	36	32	31	31	27
6-3	36	32	32	31	28
6-10	36	32	31	31	27
2-5		35	34	33	28
7-8			31	30	27
1-5				30	30
2-10					28
P(0.8)	0.2922	0.2017	0.0368	0.0132	0
Cost	24409.19	25544.48	26611.62	27746.91	29024.62

Appendix B1

Design approach : pure performance approach

Augmentation mechanism : capacity augmentation

Conf.	Initial	TOP3 aug 1-3, 3-1, 6-5	TOP6 aug 5-6, 3-5, 5-3	TOP9 aug 5-8, 3-4, 4-3	TOP12 aug 2-1, 8-5, 8-9
Fully op.	57	49	41	37	33
1-2	1290657	1222216	1178465	1178461	1178459
1-3	1.206e7	1.162e7	1.079e7	1.079e7	1.077e7
2-6	945	480	462	458	246
3-4	1477054	1477045	1477038	1477037	1477034
3-5	8722924	8661020	8632636	8630741	8630732
4-5	479	471	459	39	35
5-6	106	100	98	92	75
5-8	1.265e7	9802304	6425508	6425506	6425503
6-7	113509	43665	116	82	54
7-10	1880191	784022	91492	91478	64413
8-9	8735316	6645757	3690247	3690244	3690240
9-10	7971638	5620005	2450911	2450906	2452276
P(0.8)	0.5510	0.5510	0.4963	0.4596	0.4122
Cost	19770.44	20718.41	21641.96	22356.53	23085.55

Conf.	TOP15 aug 9-8, 1-2, 3-1	TOP18 aug 3-5, 5-3, 5-8	TOP21 aug 6-5, 5-6, 1-3	TOP24 aug 8-5, 3-5, 9-10	TOP27 aug 8-9, 5-3, 3-1
Fully op.	31	28	25	24	23
1-2	1178458	1178457	1178437	1178435	1178434
1-3	1.077e7	1.077e7	1.064e7	1.064e7	1.064e7
2-6	37	34	28	27	25
3-4	1477031	1477029	1477026	1477025	1477024
3-5	8630728	8630727	8630680	8630680	8630679
4-5	33	30	26	25	23
5-6	51	49	48	44	43
5-8	6425500	6425500	5333503	5333502	5333501
6-7	43	39	35	30	29
7-10	51744	51742	25486	25484	25483
8-9	3690232	3690232	2941813	2941813	2941811
9-10	2448177	2448176	1672429	1672429	1672427
P(0.8)	0.3685	0.3685	0.3685	0.3685	0.3685
Cost	23915.97	24792.09	25728.08	26487.09	27395.85

Appendix B2

Design approach : a) a performance-based choice of state

b) a performability-based choice of topology

Augmentation mechanism : capacity augmentation

Conf.	Initial	TOP3 aug 2-6, 6-2, 6-7	TOP6 aug 7-6, 5-6, 6-5	TOP9 aug 5-4, 6-7, 4-5	TOP12 aug 6-2, 7-6, 5-6
Fully op.	57	47	37	32	28
1-2	1290657	48048	61	56	47
1-3	1.206e7	1890831	900174	900126	344867
2-6	945	944	924	914	911
3-4	1477054	1477045	1477034	54	50
3-5	8722924	2103821	1918824	237375	146415
4-5	479	466	455	455	450
5-6	106	77	63	57	54
5-8	1.265e7	5169593	1250545	849530	367516
6-7	113509	110941	120	115	107
7-10	1880191	1098855	64991	64985	51762
8-9	8735316	3338098	502381	284533	42358
9-10	7971638	3482691	506756	428871	146142
P(0.8)	0.5510	0.5510	0.4048	0.3150	0.3150
Cost	19770.44	20782.43	21638.82	22494.85	23447.04

Conf.	TOP15 aug 7-10, 2-6, 10-7	TOP18 aug 6-5, 6-7, 6-2	TOP21 aug 7-6, 2-1, 1-2	TOP24 aug 7-10, 2-6, 3-1	TOP27 aug 4-3, 3-4, 1-3
Fully op.	26	23	21	20	19
1-2	34	27	27	25	24
1-3	47731	14654	1395	40	39
2-6	910	906	490	260	33
3-4	47	45	43	42	41
3-5	23743	23739	616	46	44
4-5	448	445	443	231	22
5-6	43	41	33	29	27
5-8	128573	36858	3403	137	135
6-7	105	94	92	90	88
7-10	51762	70	55	52	48
8-9	42111	136	43	39	37
9-10	146126	12296	55	48	45
P(0.8)	0.3150	0.2134	0.1516	0.0709	0
Cost	24448.49	25412.68	26147.33	27175.94	27985.93

Appendix B3

Design approach : a) a performability-based choice of state

b) a performability-based choice of topology

Augmentation mechanism : capacity augmentation

Conf.	Initial	TOP3 aug 6-2, 2-6, 2-6	TOP6 aug 6-7, 7-6, 6-5	TOP9 aug 5-4, 5-6, 4-5	TOP12 aug 6-7, 6-2, 7-6
Fully op.	57	49	39	32	29
1-2	1290657	112253	178	48	43
1-3	1.206e7	2009847	806513	603037	50757
2-6	945	945	935	914	913
3-4	1477054	1477046	1477036	53	50
3-5	8722924	1893801	1371914	114657	23885
4-5	479	468	458	454	450
5-6	106	89	61	57	53
5-8	1.265e7	6844506	1922407	1215836	446734
6-7	113509	110940	43676	115	115
7-10	1880191	1098887	488205	64984	64979
8-9	8735316	4504603	1330124	504967	116919
9-10	7971638	4132604	1698390	507672	404323
P(0.8)	0.5510	0.5510	0.5070	0.3150	0.3150
Cost	19770.44	20897.85	21722.62	22604.27	23524.84

Conf.	TOP15 aug 1-2, 6-5, 6-7	TOP18 aug 2-1, 3-1, 7-10	TOP21 aug 1-3, 5-6, 2-6	TOP24 aug 6-2, 10-7, 7-6	TOP27 aug 7-10, 3-4, 4-3
Fully op.	25	23	21	20	19
1-2	36	34	27	25	24
1-3	50710	37454	33117	41	39
2-6	703	264	35	34	33
3-4	47	45	43	42	41
3-5	5045	669	96	47	44
4-5	447	445	443	441	22
5-6	48	34	32	29	27
5-8	275810	141964	141910	3400	135
6-7	100	94	90	89	88
7-10	89	62	50	49	48
8-9	393	146	69	40	37
9-10	15465	12837	51	47	45
P(0.8)	0.2975	0.2134	0.0709	0.0372	0
Cost	24332.36	25198.30	26212.09	27185.10	27985.93

Appendix B4

Design approach : a) a performance-based choice of state

b) a performance-based choice of topology

Augmentation mechanism : capacity augmentation

Conf.	Initial	TOP3 aug 5-6, 6-2, 6-7	TOP6 aug 2-6, 7-6, 6-5	TOP9 aug 4-5, 6-7, 5-4	TOP12 aug 6-2, 7-6, 7-10
Fully op.	57	47	37	32	29
1-2	1290657	686384	61	56	47
1-3	1.206e7	5084319	900174	900126	347825
2-6	945	935	924	914	913
3-4	1477054	1477045	1477034	54	51
3-5	8722924	5466820	1918824	237375	146418
4-5	479	468	455	455	452
5-6	106	91	63	57	49
5-8	1.265e7	4672168	1250545	849530	347600
6-7	113509	67386	120	115	114
7-10	1880191	670300	64991	64985	64980
8-9	8735316	2654313	502381	284533	168004
9-10	7971638	2346514	506756	428871	404495
P(0.8)	0.5510	0.5510	0.4048	0.3150	0.3150
Cost	19770.44	20704.63	21638.82	22494.85	23455.87

Conf.	TOP15 aug 5-6, 2-6, 10-7	TOP18 aug 6-5, 6-7, 6-2	TOP21 aug 7-6, 4-5, 2-1	TOP24 aug 5-6, 7-10, 2-6	TOP27 aug 3-1, 1-3, 3-4
Fully op.	26	23	21	20	19
1-2	34	27	26	23	22
1-3	47731	14654	1399	39	39
2-6	910	906	696	694	238
3-4	47	45	35	34	33
3-5	23743	23739	68	60	42
4-5	448	445	444	442	231
5-6	43	41	37	35	33
5-8	128573	36858	3421	136	134
6-7	105	94	92	89	87
7-10	51762	70	58	51	47
8-9	42111	136	102	37	36
9-10	146126	12296	12168	46	43
P(0.8)	0.3150	0.2134	0.1974	0.0709	0.0094
Cost	24448.49	25412.68	26217.02	27227.65	28121.62

Appendix C1

Design approach : a) performance-based state selection and

b) performability-based topology selection

Augmentation mechanisms : increase in connectivity and capacity augmentation

Network designed corresponds to **network A** in Section 5.12.

Conf.	Initial	TOP1 add(2-5)	TOP2 add(2-7)	TOP3 add(1-10)	TOP4 add(4-7)	TOP5 aug 1-10	TOP6 aug(7-10)
Fully op.	57	48	41	37	32	31	30
1-2	1290657	86	60	54	42	41	38
1-3	1.206e7	1783170	116836	93	78	71	68
2-6	945	71	47	41	35	34	32
3-4	1477054	1477045	1477038	1477034	80	78	75
3-5	8722924	2744017	1015026	33598	85	64	62
4-5	479	466	459	455	34	33	31
5-6	106	71	52	47	37	36	34
5-8	1.265e7	1.262e7	3044219	378520	5387	196	111
6-7	113509	145	73	66	60	59	58
7-10	1880191	1457467	110985	110946	101	99	99
8-9	8735316	8582203	1612284	106817	108	68	67
9-10	7971638	7447911	1611313	116346	128	92	91
2-5		57	46	42	35	34	32
2-7			48	40	35	34	32
1-10				41	34	33	32
4-7					37	36	34
Cost	19770.44	20905.74	21677.28	23259.63	24707.25	25503.43	26125.07

Appendix C2

Design approach : a) performance-based state selection and

b) performability-based topology selection.

Augmentation mechanisms : increase in connectivity and capacity augmentation

Traffic demands are lower than demands used in generation of results in Appendix 5.9.

Network designed corresponds to **network B** in Section 5.12.

Conf.	Initial	TOP1 add (5-10))	TOP2 add (2-5)	TOP3 aug (5-10)
Fully op.	41	36	30	26
1-2	8372	8366	33	30
1-3	2696621	2696073	168612	168612
2-6	752	744	37	32
3-4	162080	162075	162068	162064
3-5	1187667	1187635	160307	160304
4-5	385	380	372	368
5-6	79	71	38	33
5-8	6978393	605515	605511	79
6-7	730	495	42	34
7-10	223568	223560	36291	36287
8-9	4514536	178144	178121	37
9-10	3002000	12058	11934	33
5-10		41	34	34
2-5			36	33
Cost	13769.06	14807.21	15942.51	16990.67

Conf.	TOP4 aug 2-1	TOP5 add (2-4)	TOP6 aug 7-6	TOP7 aug 6-5
Fully op.	25	24	21	19
1-2	30	26	22	21
1-3	58845	39	33	32
2-6	32	30	26	23
3-4	162064	28	22	21
3-5	44218	22424	40	39
4-5	367	27	26	25
5-6	32	29	26	26
5-8	78	77	46	43
6-7	33	30	27	24
7-10	36284	36279	2940	75
8-9	37	36	30	28
9-10	32	31	27	25
5-10	34	32	28	26
2-5	31	28	23	22
2-4		25	22	20
Cost	17219.80	18407.94	19251.98	19547.98

Appendix C3

Design approach : a) performance-based state selection and

b) performability-based topology selection.

Augmentation mechanisms : increase in connectivity and capacity augmentation

Traffic demands are higher than demands used in generation of results in Appendix 5.9.

Network designed corresponds to **network C** in Section 5.12.

Conf.	Initial	TOP1 aug (2-6)	TOP2 add(4-10)	TOP3 add(2-10)	TOP4 aug (2-6)	TOP5 aug (6-7)
Fully op	114	85	74	61	57	52
1-2	3.53e7	7189661	7178408	2505485	1254977	467819
1-3	1.05e8	2.51e7	2.51e7	1.27e7	6720932	4687133
2-6	1191	1191	1144	102	102	99
3-4	2.34e7	2.34e7	6065721	6065703	6065699	6065692
3-5	7.43e7	2.65e7	1.58e7	9477004	4830142	4037560
4-5	628	591	79	66	62	57
5-6	174	122	106	74	67	58
5-8	7.98e7	3.29e7	1.82e7	1.05e7	6773346	4621707
6-7	1466824	246518	245348	86610	33889	33889
7-10	1.95e7	5890189	5890142	5585246	2363105	1707315
8-9	7.60e7	3.10e7	1.70e7	9746058	6239238	4380350
9-10	5.23e7	1.99e7	1.01e7	5714181	3386006	2770603
4-10			85	71	65	61
2-10				74	67	58
Cost	40337.48	41085.09	42646.96	43924.66	44684.26	45231.04

Conf.	TOP6 aug(4-10)	TOP7 aug(2-10)	TOP8 aug(2-10)	TOP9 aug(4-10)	TOP10 aug(2-6)	TOP11 add(1-7)
Fully op	46	41	38	35	34	33
1-2	467408	27156	152	130	127	117
1-3	4655234	1775436	551785	545239	256902	59906
2-6	88	65	57	53	53	44
3-4	944060	944054	944052	112568	112567	112564
3-5	1861438	856553	330668	71459	14101	749
4-5	46	42	39	35	34	33
5-6	50	44	41	37	36	34
5-8	1891181	749982	319988	39652	35520	337
6-7	33811	228	69	66	59	57
7-10	1707301	1503279	1417031	1417029	524207	127828
8-9	1714180	801773	449006	116015	115694	14603
9-10	1098350	779177	653277	280506	280416	136992
4-10	61	55	52	52	50	48
2-10	51	51	51	47	45	43
1-7						34
Cost	46802.91	48090.61	49390.31	50974.18	51745.78	52888.67

Conf.	TOP12 aug 6-5	TOP13 aug 4-10	TOP14 aug 7-1	TOP15 aug 10-2	TOP16 aug 6-5	TOP17 aug 4-10	TOP18 aug 10-7
Fully op	32	31	30	29	29	28	28
1-2	80	78	75	57	50	49	48
1-3	59743	59704	23658	209	199	192	118
2-6	41	40	38	37	36	35	34
3-4	112563	533	532	531	529	59	58
3-5	168	112	104	101	87	75	74
4-5	32	31	30	29	29	28	28
5-6	34	33	32	32	32	31	30
5-8	307	106	90	86	83	67	67
6-7	46	46	45	41	37	37	36
7-10	20237	20237	97	94	65	64	64
8-9	1213	184	112	105	93	77	76
9-10	36987	36957	2591	2584	124	120	120
4-10	46	46	45	45	44	44	43
2-10	40	40	39	39	37	37	36
1-7	33	32	32	31	30	30	30
Cost	53184.67	53982.60	54559.05	55214.90	55516.90	56320.83	56637.66

Appendix D1

Design approach : pure performance approach

Augmentation mechanism : increase in connectivity

Performance in intermediate states of failure is also shown; although it does not affect the design iterations, it does affect the performability of the network.

Conf.	Initial	TOP1 add (3-6)	TOP2 add (3-7)	TOP3 add (2-3)	TOP4 add(3-10)	TOP5 add(3-9)	TOP6 add (3-8)
Fully op	57	49	44	38	36	36	36
1-2 half	129	107	96	41	39	39	39
1-2	1290657	1178478	1178470	64	63	63	62
1-3 half	247637	247547	247531	92	90	88	88
1-3	1.206e7	1.085e7	1.085e7	1780830	1780826	1780824	1780823
2-6 half	75	65	60	43	40	39	39
2-6	945	923	914	47	44	43	46
3-4 half	144	137	132	126	124	124	124
3-4	1477054	1477047	1477043	1477037	1477035	1477034	1477034
3-5 half	66831	108	59	55	44	40	38
3-5	8722924	2866387	476677	476606	1691	87	57
4-5 half	64	56	50	45	43	42	42
4-5	479	468	461	456	454	453	453
5-6 half	71	52	47	40	38	38	37
5-6	106	56	49	42	39	38	38
5-8 half	56096	49248	81	78	49	42	40
5-8	1.265e7	7230245	1862172	1862171	256173	235	64
6-7 half	89	62	53	44	41	40	40
6-7	113509	136	79	57	50	48	47
7-10half	235	64	50	43	40	40	40
7-10	1880191	122799	114	86	70	67	66
8-9 half	19299	1037	57	53	42	39	39

Conf.	Initial	TOP1 add (3-6)	TOP2 add (3-7)	TOP3 add (2-3)	TOP4 add(3-10)	TOP5 add(3-9)	TOP6 add (3-8)
8-9	8735316	4217697	662627	662609	25292	68	68
9-10half	20436	82	54	49	41	41	41
9-10	7971638	3026475	245023	244436	128	128	127
3-6 half		53	47	41	38	38	38
3-6		57	50	42	40	39	39
3-7 half			47	42	38	38	37
3-7			49	44	40	38	38
2-3 half				41	39	39	39
2-3				44	42	41	41
3-10half					38	37	37
3-10					38	37	37
3-9 half						36	36
3-9						36	36
3-8 half							36
3-8							36
P(0.8)	0.4388	0.2797	0.1127	0.0380	0.0266	0.0073	0.0068
Cost	19770.44	20406.70	21468.70	22210.63	23510.89	24736.31	25666.14

Appendix D2

Design approach : a) performance-based choice of state and

b) performability-based choice of topology.

Augmentation mechanisms : increase in connectivity and capacity augmentation

Intermediate states of failure are considered.

Conf.	Initial	TOP1 add(1-7)	TOP2 add(2-10)	TOP3 add(4-7)	TOP4 add(1-6)	TOP5 aug 4-7	TOP6 aug 2-10
Fully op.	57	48	39	34	31	30	29
1-2 half	129	117	47	41	36	35	31
1-2	1290657	1288910	85	76	61	60	42
1-3 half	247637	114	56	47	44	42	38
1-3	1.206e7	1955852	106852	103395	179	177	154
2-6 half	75	54	41	35	33	31	30
2-6	945	468	45	38	34	33	30
3-4 half	144	135	127	39	36	34	32
3-4	1477054	1477046	1477037	79	79	71	59
3-5 half	66831	108	54	42	37	36	33
3-5	8722924	2551987	472013	1746	99	91	72
4-5 half	64	55	46	35	33	32	30
4-5	479	467	457	36	34	32	30
5-6 half	71	60	45	38	34	33	31
5-6	106	88	51	42	36	34	32
5-8 half	56096	98	53	41	38	37	34
5-8	1.265e7	3032127	423762	5406	5352	5278	188
6-7 half	89	78	45	40	35	34	32
6-7	113509	110541	82	75	42	42	37
7-10 half	235	72	49	40	37	36	34
7-10	1880191	199046	43864	74	64	59	49
8-9 half	19299	78	51	40	36	34	33

Conf.	Initial	TOP1 add(1-7)	TOP2 add(2-10)	TOP3 add(4-7)	TOP4 add(1-6)	TOP5 aug 4-7	TOP6 aug 2-10
8-9	8735316	1583312	259629	253	102	72	67
9-10 half	20436	78	67	44	41	37	37
9-10	7971638	1489547	701012	27827	7746	174	172
1-7 half		52	42	37	33	32	30
1-7		57	44	37	34	33	31
2-10 half			44	38	35	34	32
2-10			48	43	38	36	36
4-7 half				36	34	33	31
4-7				39	36	36	35
1-6 half					33	32	30
1-6					34	33	31
P(0.8)	0.4388	0.0764	0.0425	0.0307	0.0112	0.0050	0.0007
Cost	19770.44	20913.33	22191.03	23638.65	24582.58	25311.39	25622.22

Appendix D3

Design approach : a) performability-based choice of state and

b) performability-based choice of topology

Augmentation mechanisms : increase in connectivity and capacity augmentation

Intermediate states of failure are considered.

Conf.	Initial	TOP1 add (1-7)	TOP2 add (3-7)	TOP3 add (2-3)	TOP4 add(1-10)	TOP5 add(4-10)
Fully op.	57	48	43	39	36	33
1-2 half	129	117	112	45	42	39
1-2	1290657	1288910	1288896	74	74	70
1-3 half	247637	114	99	48	41	38
1-3	1.206e7	1955852	1772797	103381	80	73
2-6 half	75	54	49	41	37	34
2-6	945	468	462	45	40	36
3-4 half	144	135	131	126	123	37
3-4	1477054	1477046	1477042	1477037	1477034	80
3-5 half	66831	108	60	56	45	39
3-5	8722924	2551987	472583	472507	1675	91
4-5 half	64	55	51	45	42	34
4-5	479	468	462	457	454	35
5-6 half	71	60	53	43	40	36
5-6	106	88	78	48	43	39
5-8 half	56096	98	58	55	44	38
5-8	1.265e7	3032127	620478	620092	5400	79
6-7 half	89	78	72	45	43	39
6-7	113509	110548	110541	81	77	73
7-10 half	235	72	53	45	41	38
7-10	1880191	199046	306	94	70	64
8-9 half	19299	78	52	49	42	36

Conf.	Initial	TOP1 add (1-7)	TOP2 add (3-7)	TOP3 add (2-3)	TOP4 add(1-10)	TOP5 add(4-10)
8-9	8735316	1583312	109006	107022	113	57
9-10 half	20436	78	53	49	43	37
9-10	7971638	1489547	117390	113000	128	63
1-7 half		52	47	41	37	34
1-7		57	50	42	38	34
3-7 half			46	41	38	33
3-7			48	43	38	34
2-3 half				42	39	35
2-3				44	41	37
1-10 half					38	34
1-10					39	35
4-10 half						34
4-10						36
P(0.8)	0.4388	0.0764	0.0725	0.0380	0.0120	0.0000
Cost	19770.44	20913.33	21975.33	22717.26	24299.61	25861.48

References

[AUP91] B. E. Aupperle and J. F. Meyer, "State space generation for degradable multiprocessor systems", Proceedings of the 21st International Symposium on fault-tolerant computing, June 1991, pages 308-315.

[BAR92] J. A. Barria von Bischhoffshausen, "Performance of data networks in fault situations", PhD Thesis, University of London, 1992.

[BER87] D. Bertsekas and R. Gallagher, *Data Networks*, Prentice-Hall International, Inc., 1987.

[CAS94] S. Casale, V. Catania, A. Puliafito and L. Vita, "A remote bridging technique to increase performability in distributed systems", IEEE Transactions on Industrial Electronics, Vol. 41, No. 2, April 1994, pages 182-190.

[CHA72] K. M. Chandy and R. A. Russell, "The design of multipoint linkages in a teleprocessing tree network", IEEE Transactions on Computers, Vol. 21, No. 10, October 1972, pages 1062-1066.

[CIC87] B. Ciciani and V. Grassi, "Performability evaluation of fault-tolerant satellite systems", IEEE Transactions on Communications, Vol. 35, No. 4, April 1987, pages 403-409.

[CIC92] B. Ciciani, "Fault-tolerance considerations for redundant binary-tree dynamic Random-Access Memory (RAM) chips", IEEE Transactions on Reliability, Vol. 41, No. 1, March 1992, pages 139-148.

[CON92] C. Constantinescu, "Predicting performability of a fault-tolerant microcomputer for process control", IEEE Transactions on Reliability, Vol. 41, No. 4, December 1992, pages 558-564.

[CON93] C. Constantinescu and C. Sandovici, "Performability evaluation of a gracefully degrading microcomputer", Computers in Industry, Vol. 22, No. 2, August 1993, pages 181-186.

[COU75] P. J. Courtois, "Error analysis in nearly-completely decomposable

stochastic systems", Econometrica, Vol. 43, No. 4, July 1975, pages 691-709.

[COU77] P. J. Courtois, *Decomposability - Queueing and Computer System Applications*, Academic Press, Inc., 1977

[COU86] P. J. Courtois and P. Semal, "Computable bounds for conditional steady-state probabilities in large Markov chains and queueing models", IEEE Journal on Selected Areas of Communications, Vol. 4, No. 6, September 1986, pages 926-937.

[DES84] E. de Souza e Silva, S. S. Lavenberg and R. R. Muntz, "A perspective of Iterative Methods for the Approximate Analysis of Closed Queueing Networks", Mathematical Computer Performance and Reliability, G. Iazeolla, P. J. Courtois and A. Hordijk (editors), Elsevier Science Publishers B.V. (North-Holland), 1984, pages 225-244.

[DES89] E. de Souza e Silva and H. R. Gail, "Calculating availability and performability measures of repairable computer system using randomization", Journal of the Association for Computing Machinery, Vol. 36, No. 1, January 1989, pages 171 - 193.

[DES92] E. de Souza e Silva and H. R. Gail, "Performability analysis of computer systems: from model specification to solution", Performance Evaluation 14, 1992, Elsevier Science Publishers B.V., pages 157-196.

[DIM89] D. D. Dimitrijevic and M. S. Chen, "An integrated algorithm for probabilistic protocol verification and evaluation", Proceedings IEEE INFOCOM '89, 1989, pages 69-73.

[DON87] L. Donatiello and B. R. Iyer, "Analysis of a composite performance reliability measure for fault-tolerant systems", Journal of the Association for Computing Machinery, Vol. 34, No. 1, January 1987, pages 179-199.

[DON91] L. Donatiello and V. Grassi, "On evaluating the cumulative performance distribution of fault-tolerant computer systems", IEEE Transactions on Computers, Vol. 40, No. 11, November 1991, pages 1301-1307.

[ESA66] L. R. Esau and K. C. Williams, "A method for approximating the optimal network", IBM System Journal, Vol. 5, No. 3, 1966, pages 142-147.

[FRA73] L. Fratta, M. Gerla and L. Kleinrock, "The Flow Deviation Method:

an approach to Store-and-Forward Communication network design", Networks 3, 1973, pages 97-133.

[FRA94] L.J.N. Franken and B.R. Haverkort, "The performability manager", IEEE Network, Vol. 8, 1994, pages 24-32.

[GER74] M. Gerla, H. Frank, W. Chou and J. Eckl, "A cut saturation algorithm for topological design of packet switched communication networks", IEEE National Telecommunications Conference, December 1974, pages 1074-1085.

[GER77] M. Gerla and L. Kleinrock, "On the topological design of distributed computer networks", IEEE Transactions on Communications, Vol. 25, No. 1, January 1977, pages 48-60.

[GER90] A. Gersht and R. Weihmayer, "Joint optimisation of data network design and facility selection", IEEE Journal on Selected Areas in Communications, Vol. 8, No. 9, December 1990, pages 1667-1681.

[GOY87] A. Goyal and A. Tantawi, "Evaluation of Performability for degradable computer systems", IEEE Transactions on Computers, Vol. 36, No. 6, June 1987, pages 738-744.

[GRA77] W. K. Grassman, "Transient solutions in Markovian queues", European journal of operational research, Vol. 1, 1977, pages 396-402.

[GRA90] W. K. Grassman, "Finding transient solutions in Markovian event systems through randomisation", First International Workshop on the numerical solution of markov chains, W. J. Stewart, Marcel Dekker, 1990, pages 357-371.

[GRA91] V. Grassi, "Cost effectiveness analysis of different fault tolerant strategies for hypercube systems", Proceedings of the 21st International Symposium on fault-tolerant computing, June 1991, pages 196-203.

[GRA92] V. Grassi and L. Donatiello, "Sensitivity Analysis of Performability", Performance Evaluation 14, 1992, Elsevier Science Publishers B. V., pages 227-237.

[HAR92] P. G. Harrison and N. M. Patel, *Performance Modelling of Communication Networks and Computer Architectures*, Addison-Wesley Publishing Company Inc., 1993.

[HEI88] P. Heidelberger and A. Goyal, "Sensitivity Analysis of Continuous Time Markov Chains Using Uniformisation", Computer Performance and Reliability, G. Iazeolla, P. J. Courtois, and O. J. Boxma (editors), Elsevier Science Publishers B. V. (North-Holland), 1988, pages 93-104.

[ISL90] S. M. R. Islam and H. H. Ammar, "Numerical solutions of Markov Reward Models using Laguerre functions", First International Workshop on numerical solutions of Markov chains, W. J. Stewart (editor), Marcel Dekker Inc., 1990, pages 645-648.

[ISL91] S. M. R. Islam and H. H. Ammar, "Performability analysis of distributed real-time systems", IEEE Transactions on Computers, Vol. 40, No. 11, November 1991, pages 1239-1251.

[ISL92] S. M. R. Islam and H. H. Ammar, "Performability of integrated software-hardware components of real-time parallel and distributed systems", IEEE Transactions on Reliability, Vol. 41, No. 3, September 1992, pages 352-362.

[IYE86] B. R. Iyer, L. Donatiello, and P. Heidelberger, "Analysis of performability for stochastic models of fault-tolerant systems", IEEE Transactions on Computers, Vol. 35, No. 10, October 1986, pages 902-907.

[KAN92] C. G. Kang and H. H.Tan, "Fault-tolerant Capacity and Flow Assignment in Packet Switched Networks", Proceedings of IEEE MILCOM 92, 1992, pages 165-171.

[KEM60] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*, D. Van Nostrand Company Inc., 1960, pages 123-125.

[KER74] A. Kershenbaum and W.Chou, "A unified algorithm for designing multidrop teleprocessing networks", IEEE Transactions on Communications, Vol. 22, No. 11, Nov. 1974, pages 1762-1772.

[KLE76] L. Kleinrock, *Queueing Systems, Vol. II: Computer applications*, John Wiley & sons, 1976.

[KRU56] J. G. Kruskal, "On the shortest spanning subtree of a graph and the travelling salesman problem", Proceedings of the American Mathematical Society, Vol. 7, 1956, pages 48-50.

[KUL86] V. Kulkarni, V. F. Nicola, R. M. Smith and K. S. Trivedi, "Numerical

evaluation of performability and job completion time in repairable fault-tolerant systems", Proceedings of the 16th International Symposium on Fault-tolerant Computing Systems, 1986, pages 252-257.

[LI84] V. O. K. Li and J. A. Silvester, "Performance analysis of networks with unreliable components", IEEE Transactions on Communications, Vol. 32, No. 10, October 1984, pages 1105-1110.

[LIN80] A. H. K. Ling, "Mathematical modelling", *Reliability and Maintainability of Electronic Systems*, J. E. Arsenault and J. A. Roberts (editors), Computer Science Press, Inc., 1980, pages 103-130.

[LOP93] N. Lopez-Benitez and K. S. Trivedi, "Multiprocessor performability analysis", IEEE Transactions on Reliability, Vol. 42, No. 4, December 1993, pages 579-587.

[MEY78] J. F. Meyer, "On evaluating the performability of degradable computing systems", Proceedings of the 8th International Conference on Fault-tolerant Computing, 1978, pages 44-49.

[MEY80] J. F. Meyer, "On evaluating the performability of degradable computing systems", IEEE Transactions on Computers, Vol. 29, No. 8, August 1980, pages 720-731.

[MEY82] J. F. Meyer, "Closed-form solutions of performability", IEEE Transactions on Computers, Vol. 31, No. 7, July 1982, pages 648-657.

[MEY92] J. F. Meyer, "Performability: a retrospective and some pointers to the future", Performance Evaluation 14, 1992, Elsevier Science Publishers B. V., pages 139-156.

[MEY95] J. F. Meyer, "Performability evaluation: where it is and what lies ahead", Proceedings of the IEEE International Computer Performance and Dependability Symposium, 1995, pages 334-343.

[MUN89] R. R. Muntz, E. de Souza s Silva and A. Goyal, "Bounding availability of repairable computer systems", IEEE Transactions on Computers, Vol. 38, No. 12, December 1989, pages 1714-1723.

[NAB96] H. Nabli and B. Sericola, "Performability Analysis: A new algorithm", IEEE Transactions on Computers, Vol. 45, No. 4, April 1996, pages 491-494.

[NAR94] Y. Narahari and N. Viswanadham, "Transient analysis of manufacturing systems performance", IEEE Transactions on Robotics and Automation, Vol. 10, No. 2, April 1994, pages 230-244.

[NIS88] T. Nishida and H. Miyahara, "Fault Tolerant Packet Switched Network Design Using Capacity Augmentation", Proceedings IEEE INFOCOM '88 1988, pages 934-940.

[PAT89] K. R. Pattipati, Y. Li and H.A.P. Blom, "On the instantaneous availability and performability evaluation of fault-tolerant computer systems", Proceedings of the IEEE International Conference of Systems, Man and Cybernetics, 1989, Vol. 1, pages 376-382.

[PAT93] K. R. Pattipati, Y. Li, and H. A. P. Blom, "A unified framework for the performability evaluation of fault-tolerant computer systems", IEEE Transactions on Computers, Vol. 42, No. 3, March 1993, pages 312-326.

[PRI57] R. C. Prim, "Shortest connection networks and some generalisations", Bell System Technical Journal, Vol. 36, November 1957, pages 1389-1401.

[PRO93] K.H. Prodomides and W.H. Sanders, "Performability evaluation of CSMA/CD & CSMA/DCR protocols under transient fault conditions", IEEE Transactions on Reliability, Vol. 42, No. 1, March 1993, pages 116-127.

[ROS83] S. M. Ross, *Stochastic Processes*, John Wiley & Sons, 1983.

[SHA90] R. L. Sharma, *Network Topology Optimization - The Art and Science of Network Design*, Van Nostrand Reinhold, 1990.

[SCH84] P. J. Schweitzer, "Aggregation Methods for large Markov Chains", Mathematical Computer Performance and Reliability, G. Iazeolla, P. J. Courtois and A. Hordijk (editors), Elsevier Science Publishers B.V. (North-Holland), 1984, pages 275-286.

[SCH77] M. Schwartz, *Computer-communication Network Design and Analysis*, Prentice-Hall Inc., 1977.

[SCH90] P. J. Schweitzer, "A survey of aggregation - disaggregation in large Markov chains", First International Workshop on numerical solutions of Markov chains, W. J. Stewart (editor), Marcel Dekker, 1990, pages 63-88.

[SIM61] H. A. Simon and A. Ando, "Aggregation of variables in dynamic systems",

Econometrica, Vol. 29, No. 2, April 1961, pages 111-138.

[SMI88] R. M. Smith, K. S. Trivedi and A. V. Ramesh, "Performability Analysis: measures, an algorithm, and a case study", IEEE Transactions on Computers, Vol. 37, No. 4, April 1988, pages 406-417.

[TAI93] A. T. Tai, J. F. Meyer and A. Avizienis, "Performability enhancement of fault-tolerant software", IEEE Transactions on Reliability, Vol. 42, No. 2, June 1993, pages 227-237.

[TAI96] A. T. Tai and J. F. Meyer, "Performability management in distributed database systems: an adaptive concurrency control protocol", Proceedings of the Fourth International Workshop on Modelling, Analysis and Simulation of Computer and Telecommunication systems, MASCOTS'96, 1996, pages 212-216.

[TRI84] K. S. Trivedi, "Reliability evaluation for fault-tolerant systems", Mathematical Computer Performance and Reliability, G. Iazeolla, P. J. Courtois and A. Hordijk (editors), Elsevier Science Publishers B. V. (North-Holland), 1984, pages 403-414.

[TRI94] K.S. Trivedi, M. Malhotra and R.M. Fricks, "Markov Reward Approach to Performability and Reliability Analysis", Proceedings of the Second International Workshop on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS'94, 1994, pages 7-11.

[VAN91] N. M. van Dijk, "Transient error bound analysis for continuous-time Markov reward structures", Performance Evaluation 13, 1991, Elsevier Science Publishers B. V., pages 147-158.

[VIS94] N. Viswanadham and R. Ram, "Composite performance-dependability analysis of cellular manufacturing systems", IEEE Transactions on Robotics and Automation, Vol. 10, No. 2, April 1994, pages 245-258.

[WEI71] H. Weisberg, "The distribution of linear combinations of order statistics from the uniform distribution", The Annals of Mathematical Statistics, 1971, Vol. 42, No. 2, pages 704-709.

[YAN89] C. L. Yang and P. Kubat, "Efficient computation of most probable states for communication networks with multimode components", IEEE Transactions

on Communications, Vol. 37, No. 5, 1989, pages 535-538.

[YOK91] T. Yokohira, M., Sugano, T. Nishida and H. Miyahara, "Fault tolerant packet-switched network design and its sensitivity", IEEE Transactions on Reliability, Vol. 40, No. 4, Oct. 1991, pages 452-459.

[BER84] D. P. Bertsekas, R. Gendron and W. K. Tsai, "Implementation of an optimal multicommodity network flow algorithm based on gradient projection and a flow formulation", Report LIDS-P-1364, Cambridge Mass., MIT Laboratory for information and decision systems, 1984.

[GAF87] E. M. Gafni and D. P. Bertsekas, "Asymptotic optimality of shortest path routing algorithms", IEEE Transactions on Information Theory, Vol. 33, No. 1, December 1987, pages 83-90.

BIBL.
LONDIN.
UNIV.