

Cognitive Packet Network for Bilateral Asymmetric Connections

Erol Gelenbe, *Fellow, IEEE* and Zarina Kazhmaganbetova

Abstract—Network testbed experimentation is useful to evaluate new protocols since it offers realism and repeatability under controllable conditions. Thus in this paper we use of a software network platform, the Cognitive Packet Network (CPN), to offer best-effort Quality of Service (QoS) to end users, to develop a new bilateral QoS differentiation between pairs of communicating nodes. In the proposed approach, each CPN edge or user node is a source and a destination at the same time, managing Uplink user originated traffic, and Downlink traffic sent back in response to the Uplink. The bilateral communication is implemented with four distinct QoS objectives that can be met between sender nodes (original source or destination). Traffic volume asymmetry between the received and sent data is used to trigger changes in QoS and the lower traffic rate requires short delay QoS, while higher traffic rate requires loss minimisation. The effectiveness of the approach is evaluated by several measurements.

Index Terms—Software Defined Networks; Network Experimentation; QoS; Uplink and Downlink Traffic; Bilateral QoS Differentiation; Cognitive Packet Network (CPN); QoS switching

I. INTRODUCTION

Industrial informatics and control systems are increasingly based on networks [18], and in addition to the primary control functions of industrial informatics systems, their overall performance is largely influenced by the network infrastructure which transfer the relevant sensor outputs to the decision units and then transfer decisions to the sub-systems points where they are used to control the system used [19]. Thus in the field of industrial informatics, experimentation with realistic network topologies is necessary [17] but difficult, whether one wishes to do this in an industrial and “production” context, or whether this is undertaken in academia.

Large networks are obviously available to network operators, however controlled experiments often require changing the operating conditions and even introducing hardware, software or additional test traffic which may lead to failures or poor system performance. Driving a system towards its performance limits is itself part of a significant experiment, which in turn can drive the system to fail. However, failures in an operational and production setting would be felt by the customer base during the period of the experiment which may

span several days, and would therefore be unacceptable from a commercial perspective.

Within an academic environment, large scale realistic experimentation is seldom conducted except for the use of tools such as Planetlab and the related MeasurementLab [16]. However it is difficult to fully control the conditions of an experiment which are conducted on third party machines such as in PlanetLab, since they may be undergoing other uses that the experimenter does not fully control, which can then introduce additional difficulties in interpreting measurements and understanding the precise experimental conditions [5], [6] which may vary during the experiment. Small network test-beds of up to one hundred or so “cheap” routers are available in some academic environments but do require a significant amount of air-conditioned space and power, and can be costly to maintain, while analytical models can be accurate for subsystems [34], [28], but they only provide rough approximations for large networks [21], [22].

Virtualization is an alternative that creates virtual nodes through Virtual LANs and tunnelling to control physical networks [9], [46], allowing experimental facilities [54], [4], [1] and the embedding of virtual networks in physical network architectures [3], [56]. Platform virtualization also creates multiple virtual routers [45]), although it may not accurately reflect the timing characteristics of physical networks [55]. An approach which provides more realistic experiments than widely used network simulators are the Software Defined Networks (SDN) [49], [8], [47] for a virtual network representation which run on a large number of virtual routers, and have their origin in the early ideas regarding Programmable Networks [20].

A. The Cognitive Packet Network (CPN)

A SDN is defined by the Open Network Foundation [49] as “The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices”. SDNs are expected to be “directly programmable, agile, managed”, so that they may offer a “global view”, and be “programmatically configured” by the network manager, with “open standards” and vendor neutrality, allowing “network control to be directly programmable”. The Cognitive Packet Network (CPN) software [38] has many of these characteristics. It offers a separation of the control plane from the forwarding plane and is directly programmable, quite compact with Linux based open standards and total hardware and vendor neutrality, and its network control is directly programmable. Previously we have illustrated the use

Manuscript received XXX, 2013; revised YYY.

Copyright (c) 2009 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

E. Gelenbe is with Department of Electrical and Electrical Engineering, Imperial College, London, UK, SW7 2AZ, e-mail: e.gelenbe@imperial.ac.uk

Z. Kazhmaganbetova is with JSC National Information Technologies, Orynbor 8, Astana, Kazakhstan, 010000, e-mail: zarina.kazhmaganbetova@nitec.kz, z.kazhmaganbetova@gmail.com

of CPN in various applications, including to evaluate denial of service (DoS) attacks [40]. Initially inspired by Active Networks [53] which are adaptive and context dependent, it allows the experimenter to configure lower level network functions through software, without the OpenFlow protocol adopted in more recent SDNs, and has been evaluated in many different experiments [30]. CPN encapsulates IP packets into “dumb packets” (DPs) and uses “smart packets” (SPs) which do not carry payload for exploring the network paths and measuring QoS, and “acknowledgement packets” (ACKs) that convey QoS and path information back to the source nodes where routing decisions are made. CPN separates the control plane from the forwarding plane and is programmable, with Linux based open standards and hardware neutrality, and manages network users with quality of service (QoS). Thus it follows the Software Defined Network (SDN) paradigm [49], [8] to construct a network representation which can then be run on hardware routers [48], and earlier work has focused on the use of a single common but possibly complex QoS metric that combines delay and loss, or just jitter, for all users [38]. QoS criteria that help to avoid denial of service attacks [40] and optime energy consumption [33] have also been considered.

B. Scope of this Paper

When two network virtual user nodes communicate across an industrial network, for instance for requesting measurement data and transferring back control decisions, or requesting downloads from a web site [44], [?] which stores relevant system observations and sending back to the requester the data, two asymmetric network users are interacting and each of them requires a different QoS criterion to be optimised. For instance, requests for downloads use very few packets but would want them to travel rapidly in the network with minimum delay, while the download of data would naturally require the data to arrive to destination with a minimum loss. The situation is further complicated when both end user nodes can have more than one role or more than one type of request and transmission. Thus this paper develops a design which allows to communicating end user nodes to alternate dynamically between two roles while offering best effort QoS. In order to address this issue, in this paper the Software Defined CPN is used for the design and then measurement of the performance of an architecture that simultaneously allows two end nodes which access a packet network, to have two *distinct* roles with two distinct and alternating QoS criteria. Such asymmetric QoS requirements are of interest for access to databases [11], [7] and web sites that deliver media [12], [50] in industrial networks. The concept is illustrated by showing how adaptive QoS routing between nodes which both act as as Uplink and Downlink servers can be implemented effectively in this manner.

C. Technical Approach

In this paper Uplink traffic travels from the source, the Uplink Sender (US), to the destination which is a Downlink Sender (DS). Applications including web accesses and downloading media content, require different QoS for the uplink

and downlink traffic. The Uplink and Downlink can be treated as distinct connections with the Source of one being the Destination of the other. When source and destination both have an uplink and downlink behaviour, both of these two “connections” may be different in terms of the QoS Goal of the data they are sending. Node A (in an A to B and vice-versa connection) may send be data to B as an Uplink, and sometimes as a Downlink responding from A to B. Thus traffic flows from A to B may have differentiated QoS, and may use distinct paths depending on whether the traffic from A to B is an Uplink or Downlink as sketched in Figure 1. In order to allow the end node to recognise which role it needs to play with regard to the traffic it is handling, a traffic based trigger is used to switch the QoS Goal that A or B are assuming as Uplink or Downlink to handle bilateral and possibly asymmetric connections.

II. DESIGN PRINCIPLES OF CPN

To develop the approach discussed in Section I-C, the CPN protocol can offer distinct QoS objectives to different users within a homogenous network with stateless routing based on reinforcement learning (RL), using neural networks [26], [36] as “oracles” for QoS routing, rather than routing tables. CPN’s virtual nodes’ (VN) communicate via virtual ports, and determine the best paths for each User packet with best effort QoS Goal. The QoS objectives for a user in CPN can be end-to-end packet delay, loss or jitter, or their combinations. It has also been used for other QoS Goals including the mitigation of Denial of Service attacks [40], energy optimization [41], [33] and economic gain through electronic sales [37]. CPN uses adaptive algorithms for path searching without routing tables. Every node in the CPN is aware about its immediate neighbours only. The paths are searched on demand, when there is a payload packet for transmission. In CPN, Smart Packets (SP) search for paths which match the pre-defined QoS Goal function in the best possible way. To achieve this, CPN utilises Random Neural Networks (RNN) in each router operating as an “oracle” for SPs, whose weights are updated via Reinforcement Learning (RL). CPN uses three types of packets: Smart Packets (SPs) that seek out paths for each individual user based on that user’s QoS requirements, acknowledgement (ACK) packets that bring back the paths that are discovered, together with their QoS value, to the source node of a user, and dumb packets (DPs) whose role is simply to carry payload. DPs are source routed, based on the path that a user will have selected using the different paths that have been received via ACKs. CPN implements a Goal-Directed Search [24], [29], [23], [2] with the help of SPs, while ordinary IP packets become CPN “dumb packets” [31].

When a source S has to send a DP to destination D with quality of service (QoS) Goal G, if it has a list of paths to D with measured values of the corresponding QoS G, it sends the DP in source-routed manner, and DP’s arriving at the destination generate an ACK that returns to S, with useful data such as the date at which the DP reached D and the path it took. The ACK returns to S along the inverse path, and the difference between the number of DPs from S to D and the

number of ACKs that from D to S are an estimate of the packet loss rate. Each ACK arriving to S also provides an estimate of the most recent round-trip delay from S to D . If S does not have a path to D , it forwards a SP to *each* neighbour, asking it to find the best next hop to D with the given QoS Goal G . The neighbours in turn will check if their local RNN [26], [36] has the answer to the question, and forward the SP to the resulting best neighbour, that will repeat the process, until either a path to D is found or D itself is found. A maximum number of hops criterion (here set to 30) is used to delete SPs that have been searching for too long, without finding D . At each step, a SP will store and carry the identity and local time each node it visits. When an SP reaches its destination, it generates an ACK that travels back to S in “source routed” mode using the path that was brought along by the corresponding SP; however any repetitions in that path are first eliminated to remove circular paths from the return path. On its way back to S , the ACK again triggers the measurement of local times at intermediate nodes, and the difference between the downstream local visit time and the upstream local visit time is used to update the local RNN with the reinforcement algorithm [30] for use when a SP again needs to be forwarded to the same D with the same QoS Goal. This procedure has also been tested with a genetic algorithm at the sources that creates new paths from previously discovered paths using the crossover operation [32].

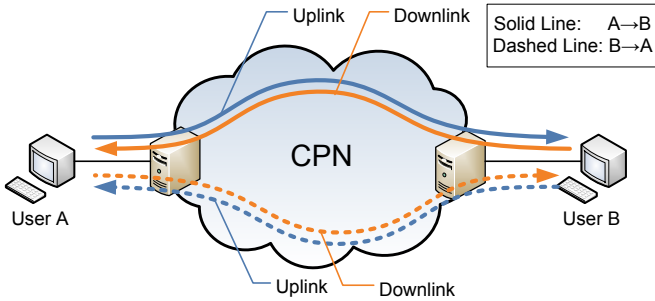


Fig. 1. The CPN network model for traffic differentiation per user.

The RNN model used by CPN is a nature inspired [39] neuronal network with spiking interaction [42] between neurons. This recurrent network model (i.e. with feedback) [35] consists of n neurons communicating with each other via positive (excitatory) and negative (or inhibitory) unit valued spikes. The neurons also receive signals from outside the network (exogenous signals), and signals can leave the network. The results of signal exchange modifies the neurons' potential levels, which are defined as non-negative integers. Each arriving positive spike increases the neuron's potential by one, and upon the arrival of a negative spike it decreases by one if the potential is positive, and otherwise has no effect. Whenever neuron i 's potential is positive, it is regarded as being “excited”, so that it fires and generates signals. This activity in turn reduces its potential with each departure reducing its potential by one. In CPN, a RNN is created in software at every intermediate CPN node for each Destination

and QoS combination since flows arriving at the node from distinct sources and headed for the same destination with the same QoS criterion, are routed from this intermediate node to the destination in the same manner. Each neuron in a given RNN is used to represent one immediate neighbour of the CPN node where it resides. In the CPN protocol each node requires information only about immediate neighbours to be able to perform routing decisions. When the next hop is to be selected for a packet with a (Source, Destination and QoS) ID set, the algorithm selects the most “excited” neuron, and sends the packet to the node that it is representing, using the steady-state probability of all the RNN neurons' integer state based on the product form solution [27], [43], which states that the stationary probability distribution of the RNN is the product of the marginal probabilities of each neuron's excitation.

CPN uses reinforcement learning (RL) to update information about which next hop to use for packets that need to reach a given destination for a given QoS Goal [38]. This RL algorithm changes the neuron weights through reward and punishment, using relevant measured QoS Goal values (e.g. delay, loss, etc.). These values are updated as follows by ACKs that return the data collected by SPs (SACKs) and/or DPs (DACKs) as they travel through the network, back to each of the intermediate nodes on a path, and also back to the source. Note that CPN uses RL to route only the SPs, while DPs are routed using “source routing” determined at the source. Thus, the ACKs that come back to the sources, bring back paths that have been identified by SPs, together with the measured QoS Goal of the path, and the source chooses among all paths that it has stored the one that for a given destination offers the best desired QoS Goal. Note also that these paths are stored in a stack with the most recently brought back path on top of the stack. As a result, older paths that may be less relevant because the network state has changed, are progressively eliminated. The reward R is inversely proportional to the QoS “Goal function” G , so that when the Goal is to minimize the delay to minimise delay we have $R \approx 1/W$, when $G = W$ is the round-trip delay that is measured from the source to the destination. More generally for QoS Goal G :

$$R = \frac{\beta}{G + \epsilon} \quad (1)$$

where $0 < \beta \leq 1$ is a proportionality constant, and $\epsilon > 0$ is a constant used to avoid division by some very small quantity. When an ACK arrives to an intermediate node, the dates it carries, namely the date at which the corresponding SP reached the destination, minus the date at which the same SP visited that same intermediate node, provides the round-trip delay, which is used to calculate the current l -th value of the reward R_l , $l = 1, 2, \dots$. The RL algorithm updates a the historical value of the reward via an exponential average to compute the threshold:

$$T_l = \alpha T_{l-1} + (1 - \alpha) R_l \quad (2)$$

where α is a constant $0 < \alpha < 1$, so that when α is greater, past values of the reward lose their importance faster. Then from (2), T_{l-1} is compared with R_l , and if $R_l > T_{l-1}$, the most recent routing decision was better than the historical

average, so that all *excitatory weights* which terminate in the previously selected neuron j are increased, while *inhibitory weights* terminating at other neurons are also increased to weaken other possible decisions.

If $R_l < T_{l-1}$, the most recent decision is not as good as the results of previous history so that the RL algorithm reduces the value of the excitation weights terminating in j and inhibitory weights terminating in j are increased, while excitatory weights to other neurons are increased and their inhibitory weights are reduced. Thus j will be less likely to be selected the next time and other neurons may be selected. Obviously, this scheme does not guarantee optimal decisions, but it pushes SPs to be routed through those paths that have better overall QoS with respect to the Goal G .

A. Bilateral traffic differentiation

For the two QoS Goals that we selected:

- The direction that carries the lower traffic direction will use delay as the QoS Goal QOS_{DELAY} , and
- Minimising packet loss QOS_{LOSS} is used for the direction that carries the larger traffic rate.

The algorithms used by CPN that we describe previously remain unchanged with regard to QOS_{DELAY} , but QOS_{LOSS} includes distinct packet loss measurements for Smart Packets, and Dumb Packets. SP loss is used only in the intermediate nodes of the CPN in order to discover and evaluate paths. DP loss is an end-to-end metric that must be measured at the sender node, since it uses the number of DPs that have been sent out to that particular destination, *minus* the number of DP ACKs that return to the sender, divided by the number of DPs sent out. In addition to the QoS decision and the path choice, each sender also manages its own flow selection.

The following enhancements were introduced to the CPN protocol to achieve bilateral traffic differentiation.

1) *Round-robin for Smart Packets*: The Uplink Server always initialises the path searching algorithm towards the Downlink Sender. DS starts to search for the path towards the US only after receiving packets from the US, but each sender needs to discover paths for every QoS Goal. SPs are marked with their QoS Identifier. For simultaneous path discovery for each QoS Goal, SPs are sent out in round-robin fashion with the different QoS identifiers: if the previous SP was marked with QoS1, the current one will be marked with QoS2 and the cycle will be repeated. Thus half of all SPs will be travelling with each of QoS1 and QoS2.

2) *Flow Selection*: . After receiving the first path for a given QoS Goal, a sender can start DP transmission. Each sender sends DPs towards the destination as Originated Dumb Packets. At the same time each sender terminates DPs from the destination; these packets are called Locally Delivered Dumb packets. Only these packets trigger the Flow Selection. The intermediate nodes are not loaded with this logic, since they just forward the DPs. A Flow Selection File stores information about packet rate in either direction (transmit TX and receive RX) per destination and user pair. The user identifier is read from the DP header **UserID** field.

3) *QoS Decision Logic*: . This algorithm selects the QoS of DPs and marks it in the DPs based on the asymmetry of traffic volume between the transmit TX and receive RX branches as described below. Before sending Originated DPs, each edge server decides which QoS criterion will be selected based on comparing the traffic rates at the TX and RX branches rates in the corresponding destination and receiver pair, and these rates are retrieved from the Flow Selection File. The proportion of TX traffic is:

$$Ratio = \frac{TXRate}{TXRate + RXRate} \quad (3)$$

and if $Ratio > 0.6$ (i.e. TX traffic exceeds 60% of overall traffic between source and destination), the QOS_{LOSS} is assigned to the DPs.

4) *Smart Packet Loss QoS in Intermediate Nodes*: . In order to implement the QOS_{DELAY} , an RNN is created in each node following the CPN protocol with a distinct RNN per combination of Source Address, Destination Address and QoS Identifier. Similarly, since other SPs are seeking out paths with minimum loss, the QOS_{LOSS} is implemented via specific RNNs at intermediate nodes for each combination of Destination Address and QoS Identifier, while the source identity can be eliminated by assigning to it the with Zero IP Address (0.0.0.0). For QOS_{LOSS} , these RNN weights are updated by RL algorithm by means of rewards, calculated by counting the number of SPs and SACKs that travel through a node, so that the loss rate at a given intermediate node is estimated with:

$$Loss_{SP} = \frac{No.SP\ Sent - No.SP\ ACK\ Recv'd}{No.SP\ Sent} \quad (4)$$

The reward is then computed with $\beta = 0.5$, $\epsilon = 0.0001$ using the expression:

$$R = \frac{\beta}{Loss_{SP} + \epsilon} \quad (5)$$

For loss measurements, the *forward* loss ratio L_f requires knowledge of the number of packets sent by the sender, and number received by the receiver. To know L_f *in real time*, the sender must receive a flow of ACKs from the receiver, but ACKs arrive to the sender *with backward loss* ratio L_b . Thus the *packet loss ratio measured by the sender* is in fact $L_m = L_f + (1 - L_f)L_b$ reported in Figures 4 and 5.

B. Dumb Packet Loss QoS with path tracking in Sender Nodes

. When a sender receives a SACK packet, it means that a new path with the less loss of SPs is available. At the sender side it is more important to have a path with the less loss of DPs. Paths should be tracked for DP Loss, but in the current CPN implementation Dumb Packets Route Register (DPRR) is always overwritten with every SACK packet that arrives. For purpose of tracking Path Control file was designed, which stores all the available paths towards the destination. The path, which resides in the DPRR, is marked in the Path Control file as a current active path, all other paths to the same destination are inactive. Each path in the file has a unique number, so that the active path ID number is written into the DP header **RouteID** field before sending. For the purpose of each path's

packet loss tracking the number of DPs and DACKs, which travel over each path, is counted in the Path Control file. The packets are distinguished and determined to have travelled a distinct path by the RouteID field value. The path's DP packet loss ratio is calculated according to the formula:

$$Loss_{DP} = \frac{DPKTs - DACKs}{DPKTs} \quad (6)$$

The path tracking algorithm decides which path will be selected for DPs travelling toward destination. When a new path is brought by SACK it is checked for uniqueness. If there is no completely the same path in the Path Control File it is immediately selected. If brought path is not genuine, it triggers the algorithm of selecting the path with less DP Loss. The DP Loss of this path that was brought by the latest SACK is calculated. At the same time the DP Loss of the current "active" path is computed to be compared with new path loss. After comparing the less loss path is selected for being written into DPRR and to become "active" in the Path Control File.

III. EXPERIMENTAL RESULTS

The modified CPN code is run on Pentiums acting as nodes with the Linux Ubuntu OS Version 2.6, connected as shown in Figure 2, 100 Mbps Ethernet links (12.5 MBps) between nodes. Experiments were run different traffic rates, and paths were identified with the list given in Table 3. Experiments were run first with with the QOS_{LOSS} to measure DP loss with one user per node and two CPN end nodes with bilateral communication. Then experiments were run with two distinct users per end node involved in bilateral connections. Each experiment was composed of 4 to 7 runs, and each run lasted approximately 3 minutes. The total number of packets for each run varied between $1 - 1.3 \times 10^5$ packets at 700 packets/sec (pps) to as much as $1.635 - 2.12 \times 10^6$ for 12×10^3 pps. Thus we report averages from experiments that ran with a substantial number of packets.

The four end users were organised in pairs, with each users communicating with a user in the other bilateral node are shown in Figure 2. Linux OS User spaces generate two distinct flows from each CPN sender node, with user *root* having User ID 0 and the second user *cpn* having User ID 1000. The first pair include *cpn* (1000) in cpn026 sending Uplink traffic at high data rate to user *root* (0) in cpn002, the latter responding at low rate with Downlink traffic. The second pair includes *cpn* (1000) in cpn002 sending a large data rate Uplink traffic to *root* (0) in cpn026, while the latter responds with low rate Downlink traffic. Thus each user in a node is assigned a QoS Goal that is different from the other user's QoS Goal in that node.

A. Experiments with One User per Sender

For one user per sender, traffic was generated from the Uplink Sender (US) 026 towards the Downlink Sender (DS) 002. Traffic from the US was much larger then from the DS, and it was first varied between 100–1000 pps with fixed packets of 1024 bytes (1 Kbyte). Then traffic was generated between 1000–12500 pps with the same packet size. In Figure

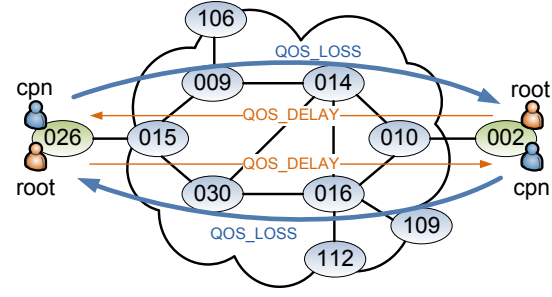


Fig. 2. The CPN Test-bed

Path ID	Route Starts at 026 Ends at 002					Length
0	015	009	014	010		Short
1	015	030	014	010		Short
2	015	030	016	010		Short
3	015	009	014	016	010	Medium
4	015	030	014	016	010	Medium
5	015	030	016	014	010	Medium
6	015	009	014	030	016	010 (L)

Fig. 3. Identifiers of Paths used in the Testbed

4 we plot the packet Loss Percentage for rates 700, 800, 900 and 1000 pps. For rates less than 700 pps, we see that the nodes were not saturated and the observed DP loss was always zero. The saturation is seen to begin slowly upwards from rate 700 pps. When CPN minimises DP Loss, sometimes there appear to be some errors, for instance a loss count can in fact result from misinterpreting a long delay in the return of a DP ACK (DACK). Nevertheless, if DACKs have long delays, the path may be saturated, and it is better to switch paths to achieve small packet loss, so that the effect is the same. The increase in packet Loss Percentage in Figure 5 shows that during the experiments the paths used saturated increasingly with increasing packet rate. We see that this trend is exacerbated above the 2000 pps packet rate, but DP loss remains low, under 0.01% , until the packet rate of 8000 pps. For a 12500 (12.5 Mbps) rate, DP Loss attains the much higher value of 0.25%. Although we may consider that the DP Loss percentage is low, we know that the network we considering is small and that it has interfaces that enjoy a large bandwidth. Obviously it is also a consequence of the good functioning of the CPN's ability to achieve the objective of QOS_{LOSS} .

B. Experiments with Two Users per Sender

With two users per sender, the effect of QOS_{LOSS} was evaluated with traffic rates in the range 1000–10000 pps, chosen to be equal at each sender, so that the traffic in the network was twice the rate generated by a user. Also, since the *root* user's traffic did not exceed 100 pps, it had a negligible effect on the results.

The measured proportion of time that a path is used for two distinct QoS criteria by each sender is summarised in Figure 6 and Figure 7. We see that the percentage of time when the same path is simultaneously used for QOS_{DELAY} and QOS_{LOSS} from nodes cpn002 and cpn026 varies in the

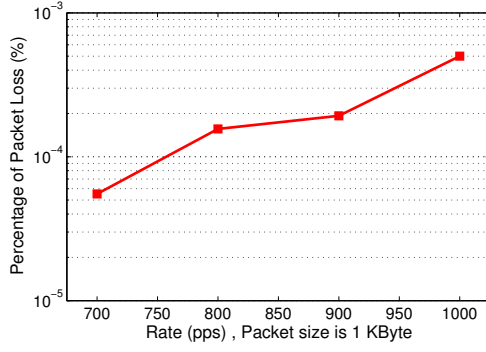


Fig. 4. Packet Loss Percentage for 700 – 1000pps

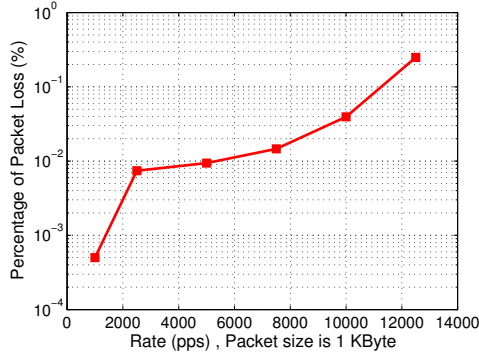


Fig. 5. Packet Loss Percentage for 1K – 12.5Kpps

small range 15 – 25%. Thus for 80% of the time, DPs from cpn002 and cpn026 travel along distinct paths, and the fluctuations for both senders are small, namely 4.5% on a measured average value 20.5% for cpn002, and roughly 3% on a measured average value 18% for cpn026.

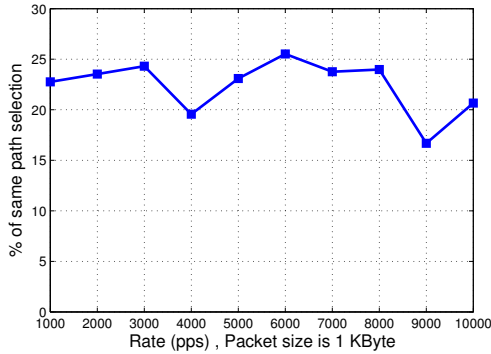


Fig. 6. Percentage of Same Path Selection by cpn002 with two Distinct QoS Criteria

The number of simultaneously used distinct paths is obviously smaller than the number of traffic flows, and measurements show that the protocol developed in this paper has a stable behaviour, with 3 paths being used more than 50% of the time, on average for 70% of the time more than three paths are used, and one path is used only 1 – 2% of the time.

Figure 8 and Figure 9 show the most frequently used paths for each CPN sender, indicating that Path_0 is the most popular for user node cpn002 with 40% average usage. Path_1

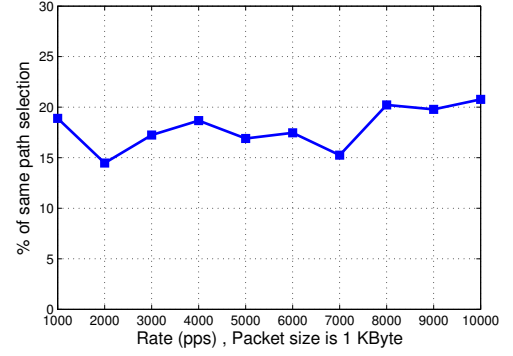


Fig. 7. Percentage of Same Path Selection by cpn026 with two Distinct QoS Criteria

and Path_2 are equally popular but twice less so than Path_0. For cpn026 Path_0 is as frequently used as Path_2 at 27–28%. Path_1 by cpn026 is three times less often used than Path_0 and Path_2, and other paths are used by both nodes almost equally at ≈ 5 -10%. We see that Path_6, as it is not at the last place in usage by both nodes, and Path_4 comes last. The paths' usage ranking for cpn026 does not seem to be sensitive to network saturation, contrary to cpn002 where at high traffic Path_0 matches Path_2 in frequency of use.

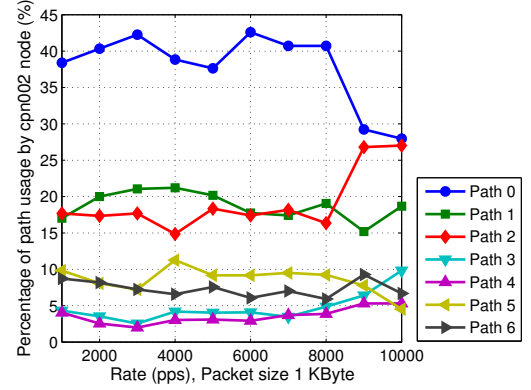


Fig. 8. Percentage of Paths Usage for sender node cpn002

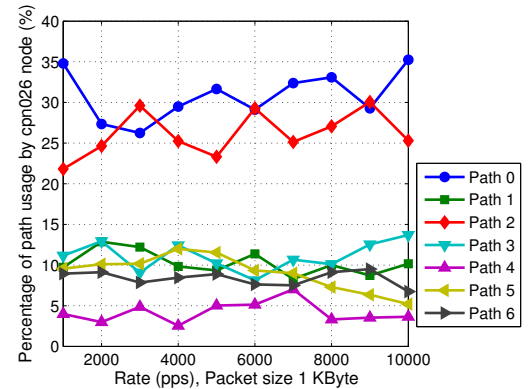


Fig. 9. Percentage Path Usage for sender node cpn026

Figures 10 and 11 show the most used paths for each of the two QoS criteria, without tracking which sender node

is forwarding the packets. The path in the reverse direction was regarded as the same path when packets in one direction travel through the same nodes as in the opposite direction. QOS_{DELAY} and QOS_{LOSS} have different preferences for distinct paths. Indeed Path_0 and Path_2 are equally used, but Path_1 is preferred by QOS_{DELAY} over QOS_{LOSS} . Thus QOS_{DELAY} uses three short paths to balance traffic, while QOS_{LOSS} uses Path_0 and Path_2. The remaining paths are equally used by each QoS criterion, except for Path_4. QOS_{LOSS} tends to to exclude Path_4 from its DP transmission, while QOS_{DELAY} uses it on average 8% of the time.

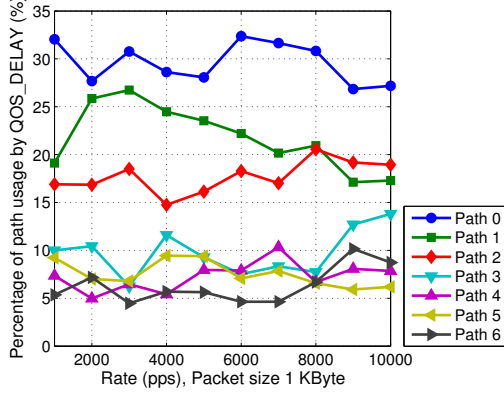


Fig. 10. Percentage Path usage for QoS DELAY

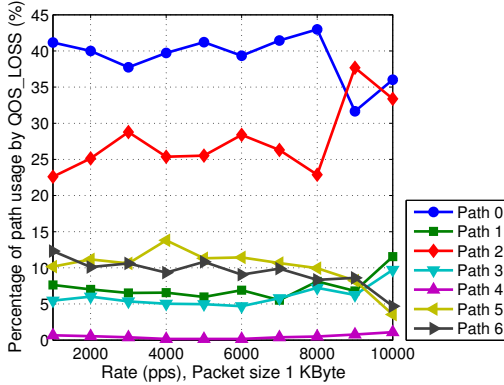


Fig. 11. Percentage Path Usage for QoS LOSS

Figure 12 shows the paths that are most used in the whole network, and the combination of path preferences from the two sender nodes with the two QoS criteria are summarised. The plots show that all paths are actually used by some proportion of the overall traffic. Path_4 is not selected by QOS_{LOSS} , but QOS_{DELAY} uses it in $\approx 8\%$ of the cases, so that its resulting usage in the network as a whole is roughly 4%. Path_1 is selected by cpn002 some 20% of the time and shares the second rank with Path_2 with regard to usage by node cpn002. We also see that cpn026 uses Path_1 only 10% of time. Thus it is not surprising that Path_1 is used in the network on average 15% of the time and therefore occupies the third place.

Finally the Figure 13 shows that as traffic rate changes the number of paths that are simultaneously used does not

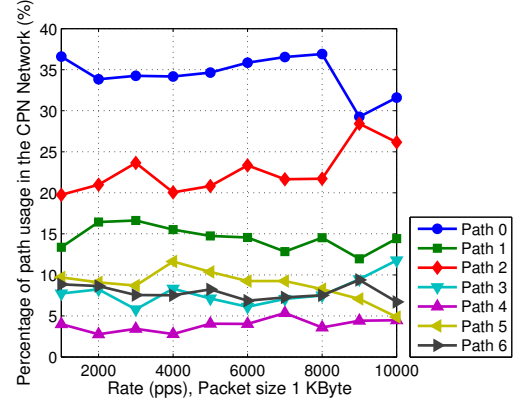


Fig. 12. Percentage of Path Usage in the Testbed

vary much. In addition, Figure 14 shows that a simple normal distribution accounts for the different number of paths being used.

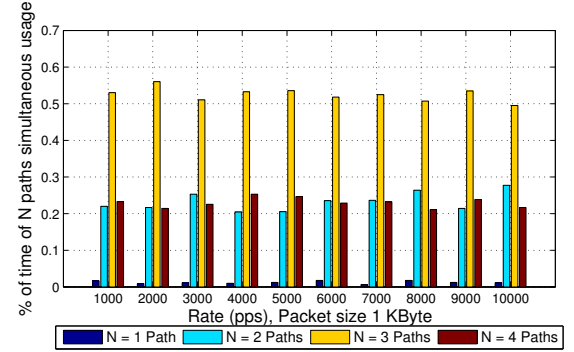


Fig. 13. Percentage of time that N paths are used versus different traffic rates.

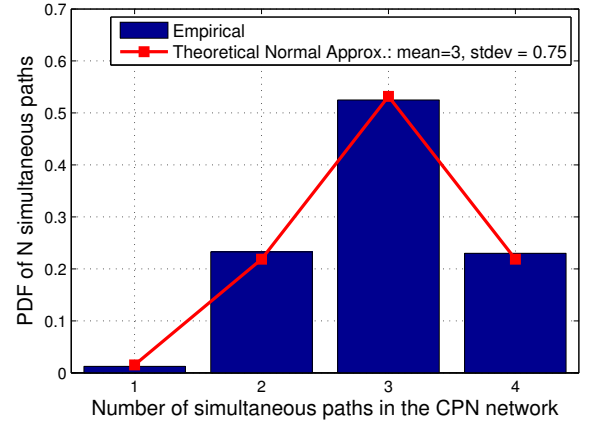


Fig. 14. The number of paths used over all traffic rates can be approximated by a Gaussian density function.

IV. CONCLUSIONS: ACHIEVEMENTS AND FUTURE WORK

Packet based network systems constitute the communication infrastructure of many industrial control and industrial systems. While such systems traditionally rely on local area networks (LAN), this infrastructure is now becoming increasingly

integrated into the Internet Protocol to facilitate the inclusion of remote data, to allow the systems to be managed across multiple sites, and also allow the control and management of very large scale industrial systems. We therefore expect that future industrial networked systems will follow technology trends by increasingly using the emerging autonomic and self-organised communication paradigms that rely on software defined networks.

Industrial systems typically have widely varying QoS requirements [52], yet the QoS offered by LANs is dominated by the medium access protocols that are being used. Thus bus based LANs such as Token Rings and Ethernet cannot offer the variety of QoS criteria [51] that can be obtained by combining admission control, flexible routing and priorities that are easily can integrated using a store-and forward network architecture. In addition, a software defined network system can make trade-offs between traffic QoS and end-user QoS, in applications that require content download and video transmission [25], [52], and can accomplish this with programmable routers that use multiple hops, alternate paths and traffic load balancing.

Thus in this paper we have combined two ideas which can advance the field of networks for industrial systems, addressing in particular the design of service oriented architectures for industrial networked systems discussed in [15], [13]. Since such systems must integrate versatile nodes that have multiple functions, we have developed a scheme that allows a network node A to communicate with another network node B using multiple paths that simultaneously support different QoS criteria. Furthermore our scheme allows both A and B to dialogue with its partner using multiple QoS criteria and multiple paths. Also, all the communicating partners can automatically switch the QoS criteria they are using based on observed traffic conditions, and this concept is in no way limited to the communication between two nodes and can be readily extended to many-to-many communications. We have then implemented and tested this conceptual system using the CPN test-bed.

To illustrate these ideas we have run experiments where end nodes A and B have dual roles of both requesting content download from the other node, and providing content download that is requested by the other node, each of these being handled with distinct QoS criteria. This experimental setting corresponds to the case of web access access for industrial networked systems which is critical for both the presentation of system status and the access to relevant data and information as discussed in [14]. We have tested the system with two useful QoS criteria, QOS_{DELAY} whose Goal is the shortest delay for the payload, and QOS_{LOSS} whose Goal is the smallest possible packet loss. To realise this new approach, the CPN protocol code was enhanced with several features, namely the Round-Robin principle for SPs that must discover paths with both of these QoS criteria, and the Flow Selection feature to tracks rates of sent and received traffic. The selection of the QoS Goal for each traffic direction was performed by the QoS Decision Logic which allocates to the larger traffic branch the QOS_{LOSS} criterion, and allocates QOS_{DELAY} to the smaller traffic rate. QOS_{LOSS} uses the SP Loss algorithm which resides in intermediate nodes, and

source routing decisions are made based on DP Loss by the sender nodes.

Experiments with one user per sender node have shown the effectiveness of the DP Loss algorithm with path tracking. For the network topology used in these experiments, the measurements showed that among seven possible paths that were used, the path that had the least DP or payload packet loss is always successfully selected. When the network was fully saturated, we still could not observe significant degradation of DP Loss, which stayed at less than 1% of the maximum interface bandwidth. Experiments also clear established effectiveness of the Bilateral Traffic Differentiation. In our experiments each of two users in one sender node had a distinct QoS criterion. The ideal case for effective CPN network resource utilisation was when four simultaneous distinct paths were used, one path for each flow. In more than 70% of the cases measured, three or four distinct viable paths could be used. In only 1% of the cases, only one path was used by all users simultaneously. Surprisingly, the path that was the least frequently used was one of medium length, rather than the longest path. Nevertheless, our measurements of path usage confirm that short paths are preferred by both of the QoS criteria. Surprisingly, the longest path was used as frequently as medium length paths, and the longest path was never the least used in all of the measurements that we conducted. This confirms, as in previous work, that path length is not the main factor for QoS routing.

The work in this paper can be extended in several directions. First, we expect to conduct experiments with different levels of background traffic so that the robustness of our proposed approach to a more varied workload can be evaluated. Secondly we plan to use of multiple QoS criteria rather than just delay and loss. More generally in future work we plan to consider a many-to-many communication system where each end user manages communications with multiple other end users, each having a bilateral and possibly distinct QoS and security [10] requirements. This corresponds to the case where the end users are all carrying out one or more distinct functions across an industrial system, so that their communications in each direction must satisfy different QoS needs. Thus each end user would maintain a table of its current correspondents with the distinct QoS requirements, and a list of different paths that are maintained and updated to best satisfy these needs.

REFERENCES

- [1] "http://www.ict-fire.eu."
- [2] O. H. Abdelrahman and E. Gelenbe, "Time and energy in team-based search," *Physical Review*, vol. E87, no. 3, p. 032125, 2013.
- [3] J. Aguilar and E. Gelenbe, "Task assignment and transaction clustering heuristics for distributed systems," *Information Sciences*, vol. 97, no. 1-2, pp. 199-219, 1997.
- [4] T. Anderson and M. K. Reiter, "Geni: Global environment for network innovations distributed services working group," 2006.
- [5] S. Basso, M. Meo, A. Servetti, and J. C. D. Martin, "Estimating packet loss rate in the access through application-level measurements," in *ACM SIGCOMM W-MUST*. ACM, 2012.
- [6] S. Bauer, D. Clark, and W. Lehr, "Understanding broadband speed measurements," in *MIT Technical Report*, 2010.
- [7] L. Borzemski and A. Kaminska-Chuchmala, "Distributed web systems performance forecasting using turning bands method," *IEEE Trans. Industrial Informatics*, vol. 9, no. 1, pp. 254-261, 2013.

- [8] J. Burt, "Software-defined networking: There's more to it than just hype," in *eWEEK*, June 2013.
- [9] M. Casado, T. Koponen, R. Ramanathan, and S. Shenker, "Virtualizing the network forwarding plane," in *Proc. of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, ser. PRESTO '10. New York, NY, USA: ACM, 2010, pp. 8:1–8:6.
- [10] M. Cheminod, L. Durante, and A. Valenzano, "Review of security issues in industrial networks," *IEEE Trans. on Industrial Informatics*, vol. 9, no. 1, pp. 277–293, January 2013.
- [11] A. Chesnais, E. Gelenbe, and I. Mitrani, "On the modelling of parallel access to shared data," *Communications of the ACM*, vol. 26, no. 3, pp. 196–202, 1983.
- [12] C. Cramer, E. Gelenbe, and H. Bakircioglu, "Low bit rate video compression with neural networks and temporal subsampling," *Proc. of the IEEE*, vol. 84, no. 10, pp. 1529–1543, October 1996.
- [13] T. Cucinotta, A. Mancina, G. Anastasi, G. Lipari, L. Mangeruca, R. Checco, and F. Rusina, "A real-time service-oriented architecture for industrial automation," *IEEE Trans. Industrial Informatics*, vol. 5, no. 3, pp. 267–277, 2009.
- [14] H. Dong and F. K. Hussain, "Focused crawling for automatic service discovery, annotation, and classification in industrial digital ecosystems," *IEEE Trans. on Industrial Electronics*, vol. 58, no. 6, pp. 2106–2116, 2011.
- [15] H. Dong, F. K. Hussain, and E. Chang, "A service search engine for the industrial digital ecosystems," *IEEE Trans. on Industrial Electronics*, vol. 58, no. 6, pp. 2183–2196, 2011.
- [16] C. Dovroli, K. Gummadi, A. Kuzmanovic, and S. D. Meinrath, "Measurement lab: Overview and an invitation to the research community," in *SIGCOMM CCR*. ACM, 2010.
- [17] FCC, "2013 measuring broadband america," in *Office of Engineering and Technology and Consumer and Governmental Affairs Bureau*. Washington, DC, USA: FCC, February 2013.
- [18] M. Felser, J. Jasperneite, and P. Gaj, "Guest editorial special section on distributed computer systems in industry," *IEEE Trans. Industrial Informatics*, vol. 9, no. 1, p. 181, 2013.
- [19] P. Gaj, J. Jasperneite, and M. Felser, "Computer communication within industrial distributed environment - a survey," *IEEE Trans. Industrial Informatics*, vol. 9, no. 1, pp. 182–189, 2013.
- [20] A. Galis, S. Denazis, C. Brou, and C. Klein, *Programmable Networks for IP Service Deployment*. Artech House Books, June 2004.
- [21] E. Gelenbe, "Probabilistic models of computer systems part i (exact results)," *Acta Informatica*, vol. 7, no. 1, pp. 35–60, 1976.
- [22] —, "Probabilistic models of computer systems part ii: Diffusion approximations: waiting times and batch arrivals," *Acta Informatica*, vol. 12, no. 4, pp. 285–303, 1979.
- [23] —, "Search in unknown random environments," *Physical Review*, vol. E82, no. 6, p. 061112, 2010.
- [24] E. Gelenbe and Y. Cao, "Autonomous search for mines," *European Journal of Operational Research*, vol. 108, no. 2, pp. 319–333, 1998.
- [25] E. Gelenbe, C. Cramer, M. Sungur, and P. Gelenbe, "Traffic and video quality in adaptive neural compression," *Multimedia Systems*, vol. 4, pp. 357–369, 1996.
- [26] E. Gelenbe and J. Fourneau, "Random neural networks with multiple classes of signals," *Neural Computation*, vol. 11, no. 4, pp. 953–963, 1999.
- [27] E. Gelenbe and K. F. Hussain, "Learning in the multiple class random neural network," *IEEE Trans. on Neural Networks*, vol. 13, no. 6, pp. 1257–1267, November 2002. [Online]. Available: <http://dx.doi.org/10.1109/TNN.2002.804228>
- [28] E. Gelenbe and R. Iasnogorodski, "A queue with server of walking type (autonomous service)," *Annales Inst. Henri Poincaré Série B*, vol. 16, no. 1, pp. 63–73, 1980.
- [29] E. Gelenbe and T. Koçak, "Area-based results for mine detection," *Geoscience and Remote Sensing, IEEE Trans. on*, vol. 38, no. 1, pp. 12–24, 2000.
- [30] E. Gelenbe, R. Lent, and A. Nunez, "Self-aware networks and QoS," *Proc. of the IEEE*, vol. 92, pp. 1478–1489, 2004.
- [31] E. Gelenbe and P. Liu, "QoS and routing in the cognitive packet network," in *Proc. of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, June 2005, pp. 517–521.
- [32] E. Gelenbe, P. Liu, and J. Lainé, "Genetic algorithms for route discovery," *IEEE Trans. Systems, Man and Cybernetics B*, vol. 36, no. 6, pp. 1247–1254, 2006.
- [33] E. Gelenbe and C. Morfopoulou, "A Framework for Energy Aware Routing in Packet Networks," *The Computer Journal*, vol. 54, no. 6, June 2011.
- [34] E. Gelenbe, "Unified approach to the evaluation of a class of replacement algorithms," *IEEE Transactions on Computers*, vol. 22, no. 6, pp. 611–618, 1973.
- [35] —, "Stability of the random neural network," *Neural Computation*, vol. 2, no. 2, pp. 239–247, 1990.
- [36] —, "The first decade of g-networks," *European Journal of Operational Research*, vol. 126, no. 2, pp. 231–232, 2000.
- [37] —, "Analysis of single and networked auctions," *ACM Trans. Internet Techn.*, vol. 9, no. 2, 2009.
- [38] —, "Steps towards self-aware networks," *Communications of the ACM*, vol. 52, no. 2, pp. 66–75, 2009.
- [39] —, "Natural computation," *The Computer Journal*, vol. 55, no. 7, pp. 848–851, 2012.
- [40] E. Gelenbe and G. Loukas, "A self-aware approach to denial of service defence," *Computer Networks*, vol. 51, no. 5, pp. 1299–1314, April 2007.
- [41] E. Gelenbe and T. Mahmoodi, "Energy-aware routing in the cognitive packet network," in *ENERGY 2011, The First International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, May 2011, pp. 7–12.
- [42] E. Gelenbe and A. Stafylopatis, "Global behavior of homogeneous random neural systems," *Applied Mathematical Modelling*, vol. 15, no. 10, pp. 534–541, 1991.
- [43] E. Gelenbe and S. Timotheou, "Random neural networks with synchronised interactions," *Neural Computation*, vol. 20, no. 9, pp. 2308–2324, 2008.
- [44] A. Jestratjew and A. Kwiecien, "Performance of http protocol in networked control systems," *IEEE Trans. on Industrial Informatics*, vol. 9, no. 1, pp. 271–276, January 2013.
- [45] X. Jiang and D. Xu, "vbet: a vm-based emulation testbed," in *Proc. of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, ser. MoMeTools '03. New York, NY, USA: ACM, 2003, pp. 95–104.
- [46] E. Keller and J. Rexford, "The "platform as a service" model for networking," in *Proc. of the 2010 internet network management conference on Research on enterprise networking*, ser. INM/WREN'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 4–4.
- [47] P. Martinez-Julia, A. F. Gómez-Skarmeta, and A. Galis, "Towards a secure network virtualization architecture for the future internet," in *Future Internet Assembly*, ser. LNCS, A. Galis and A. Gavras, Eds., vol. 7858. Springer, 2013, pp. 141–152.
- [48] M. R. Nascimento, C. Rothenberg, M. Salvador, C. N. A. Correia, S. C. de Lucena, and M. F. Magalhaes, "Virtual routers as a service: the routeflow approach leveraging software-defined networks," in *Proc. of the 6th International Conference on Future Internet Technologies*, ser. CFI '11. New York, NY, USA: ACM, 2011, pp. 34–37.
- [49] Open-Networking-Foundation, "Software-defined networking (sdn) definition," in <https://www.opennetworking.org/sdn-resources/sdn-definition>, 2013.
- [50] R. Santos, P. Pedreiras, and L. Almeida, "Demonstrating an enhanced ethernet switch supporting video sensing with dynamic qos," in *DCOSS*, 2012, pp. 293–294.
- [51] J. Silvestre-Blanes, L. Almeida, R. Marau, and P. Pedreiras, "Online qos management for multimedia real-time transmission in industrial networks," *IEEE Trans. on Industrial Electronics*, vol. 58, no. 3, pp. 1061–1071, 2011.
- [52] S. Soucek and T. Sauter, "Quality of service concerns in ip-based control systems," *IEEE Trans. on Industrial Electronics*, vol. 51, no. 6, pp. 1249–1258, 2004.
- [53] D. L. Tennenhouse and D. Wetherall, "Towards an active network architecture," *Computer Communication Review*, vol. 37, no. 5, pp. 81–94, 2007.
- [54] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*. Boston, MA: USENIX Association, Dec. 2002, pp. 255–270.
- [55] J. Whiteaker, F. Schneider, and R. Teixeira, "Explaining packet delays under virtualization," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 38–44, 2011.
- [56] Y. Xin, I. Baldine, A. Mandal, C. Heermann, J. Chase, and A. Yumerefendi, "Embedding virtual topologies in networked clouds," in *Proc. of the 6th International Conference on Future Internet Technologies*, ser. CFI '11. New York, NY, USA: ACM, 2011, pp. 26–29.



Erol Gelenbe is the Professor in Computer Communications at Imperial College London and he currently holds the Dennis Gabor Chair in the Department of Electrical and Electronic Engineering. A Life Fellow of the IEEE, he graduated from the Middle East Technical University (Ankara) and holds a Master's and PhD degree from the Polytechnic Institute of New York University. He is a Member of the French National Academy of Engineering, and of the Science Academies of Hungary, Poland and Turkey. Known for inventing the G-Network

queueing network models and the Random Neural Network Model and their product form solutions, he is one of the designers of the QNAP performance modeling package and the FLEXSIM object-oriented Flexible Manufacturing System simulation method. His other practical developments include the first random access fiber optics network (XANTHOS) and the multiprocessor voice packet based switch SYCOMORE.



Zarina Kazhmaganbetova is currently with JSC National Information Technologies at the position of Analyst, Research Associate in the Laboratory of Cloud Technologies in Astana, Kazakstan. She obtained her Bachelor's degree from Almaty Institute of Power Engineering and Telecommunications and her Master's degree from Imperial College London. Before starting at Imperial College London she had already possessed 6 years of industrial experience in the field of Wireless Mobile Communications. She worked at the engineering positions in the Mobile

Operator Companies of Kazakhstan and at the position of Technical Support Engineer in the International Telecommunication Vendor Company - Alcatel-Lucent.