Real-Time Traffic over the Cognitive Packet Network

Lan Wang and Erol Gelenbe

Intelligent Systems and Networks Group
Department of Electrical and Electronic Engineering
Imperial College London
{lan.wang12,e.gelenbe}@imperial.ac.uk

Abstract. Real-Time services over IP (RTIP) have been increasingly significant due to the convergence of data networks worldwide around the IP standard, and the popularisation of the Internet. Real-Time applications have has strict Quality of Service (QoS) constraint, which poses a major challenge to IP networks. The Cognitive Packet Network (CPN) has been designed as a QoS-driven protocol that addresses user-oriented QoS demands by adaptively routing packets based on online sensing and measurement, and in this paper we design and experimentally evaluate the"Real-Time (RT) over CPN" protocol which uses QoS goals that match the needs of real-time packet delivery in the presence of other background traffic under varied traffic conditions. The resulting design is evaluated via measurements of packet delay, delay variation (jitter) and packet loss ratio.

Keywords: Cognitive Packet Network, real-time over CPN.

1 Introduction

Communication services such as telephone, broadband and TV are increasingly migrating into the IP network with the worldwide deployment and operation of Next-Generation Networks which consolidate mobile and data networks into common IP infrastructures [1, 2]. Applications such as remote security [3] and the needs of cyber-physical systems [4] are also pushing the bounds in this direction. Although the convergence of network services enables the transport of real-time, video, voice and data traffic over IP networks which were originally designed to offer best-effort services for non-real-time data traffic, our IP networks are not well designed to guarantee Quality of Service (QoS) for such diverse communications [5], including real-time (RTIP), voice (VOIP) [6] and web based search, video and multimedia [7–10]. Real-time constraints [11] could be well assured in dedicated circuit-switched connections, but are difficult to cover with IP networks [12].

Furthermore, industrial distributed systems and industrial networks [13–16] are pushing the curve with respect to real-time communication needs. Real-time needs for enhanced reality and telepresence [17] are also creating specific needs in

on-line QoS management for multimedia systems [18]. Thus in the past decade, many QoS approaches have been proposed including IntServ & RSVP, DiffServ and MPLS, which specify a coarse-grained mechanism for providing QoS, but fully satisfactory results are not yet available in this area and there is space for further investigation and research [19–21].

Thus this paper investigates the use of the Cognitive Packet Network (CPN) routing protocol for real-time traffic. CPN is a QoS-driven routing protocol based on a smart network where users (applications) can declare their desired QoS requirements, while CPN adaptively routes their traffic by means of online sensing and measurement so as to provide the best possible QoS requested by the users [22, 23], as a means to support and enhance the users' QoS needs. To address the needs of real-time traffic, this paper introduces a variant of CPN with a "goal" function that addresses real-time needs with regard to "Jitter" according to RFC3393 [24], and implements new functions in CPN to support multiple QoS classes for multiple traffic flows [25]. Furthermore, since adaptive routing takes place when CPN is used in order to meet some of the requirements of this QoS goal, path switching can occur and we evaluate the correlation of end-to-end packet loss with path switching and with the delay that occurs at the end-user's re-sequencing buffer that is needed to insure that packets are received in time-stamp order. The main experimental result of this paper is that using Jitter Minimisation as the QoS goal for routing all of the traffic flows in the network, we can minimize jitter but also minimize delay and loss for real-time traffic. However we also see that adaptive schemes that switch paths to minimize delay, loss or jitter, may also cause buffer overflow and hence losses in the output re-sequencing buffers that deliver packets in time-stamp order for real-time traffic.

1.1 The Cognitive Packet Network Protocol

In CPN, QoS requirements specified by users, such as *Delay*, *Loss*, *Energy Consumption*, or a combination of the above, are incorporated in the "goal" function which is used for the CPN routing algorithm. Three types of packets are used by CPN: smart packets (SPs), dumb packets (DPs) and acknowledgments (ACKs). SPs explore the route for DPs and collect measurements; DPs carry payload and also conduct measurements; ACKs bring back the information that has been discovered by the SPs and DPs including route information and measurement results. SPs discover the route using random neural network (RNN)-based reinforcement learning (RL) [26, 27] which resides in each node. Each RNN in a node corresponds to a QoS class and destination pair and each neuron of a RNN represents the choice to forward a given packet over a specific outgoing link from the node. The arrival of an SP for a specific QoS class at a node triggers the execution of the RNN-based algorithm.

During this process, the weights of the corresponding RNN are updated by Reinforcement Learning (RL) using QoS goal-based measurements that are collected by SPs and brought back by ACKs and stored in a mailbox at the node.

Following that, the output link corresponding to the most excited neuron is chosen as the routing decision. More detail on CPN routing is available in [22]. The paths explored by the SPs for the desired QoS goal are brought back to the source node by an ACK packet and stored in a list of possible paths. Among these, DPs will select one which is recent and has the best QoS goal; as DPs are sent forward, they will also create returning ACKs which can be used to update the performance (e.g. end-to-end delay or jitter) that is needed by the application or user that is forwarding the packets.

1.2 CPN for Real-Time Traffic

Previous research has suggested the ability of CPN to provide improved QoS for real-time traffic compared with the conventional Internet protocol such as OSPF [28]. This paper presents an experimental study specifically for Real-Time service over CPN, the principal building block of which is shown in Fig. 1.

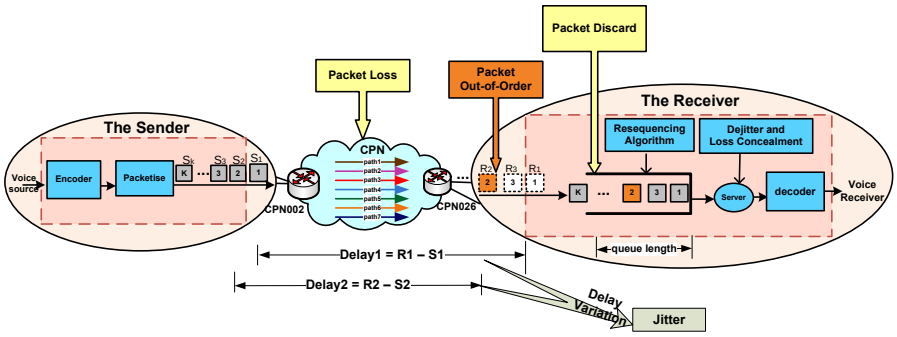

Fig. 1. RTIP System Structure [29]

In Real-Time services that are important in a variety of applications, including industrial control, media transmission and real-time, the traffic sent from a source must not only arrive with a small delay, but it must also arrive with small variance not to disrupt the needs of the control application or the media receiver. However, very importantly, the end receiver must receive the sender's packets *in the order in which they are sent*.

Indeed in most control applications the order in which the control messages are received has great importance for industrial control [30, 31]. Similarly, any measurements that are sent to a controller from different parts of a distributed system, have to be received in time-stamp sending order. This is similar for media where video must arrive in the order that was prescribed by the sender, and for real-time conversations where the meaning of a conversation cannot be preserved if the packets arrive in some other than time-stamp order. Note that for data packets in file transfers, this aspect is of no great importance since the pacjkets

can be reassembled in a receiving file system. Because of the human delays at the terminal, email also does not have such stringent constraints on the order of packet arrival. Thus, at the sender which resides in a RTIP application installed at a CPN node, the original analog measurement signals are sampled with a fixed frequency, which is commonly 8000 Hz if real-time standards are being used, and then each sample is encoded by using using different standards (e.g. G.711, G.729, G.729a, G.723.1, G.726, G.722, G.728, for audio) which differ in the compression algorithms and the resulting bitstream bandwidth. The encoded bitstream is packetised into IP packets by adding RTP header, UDP header, and IP header.

These packets can then be transmitted across the IP network employing the CPN protocol, where IP-CPN conversion is performed at the source node by encapsulating IP packets into CPN packets which are routed based on their QoS requirements. Due to the shared nature of the IP network, real-time traffic may undergo transmission impairments including delay, jitter, packet de-sequencing and packet loss. CPN can alleviate these impairments by smartly selecting the path that provides the best possible QoS required by the user (or an application) [32] and thus the packets in a real-time traffic flow may traverse the CPN network successively along several different paths. At the receiver, packets are queued in a buffer called the re-sequencing buffer. The purpose of the buffer is to reorder packets and buffer them to reduce jitter. Packets that arrive later than the maximum time allowed for the real-time signal recovery, or those that provoke buffer overflow, are discarded, contributing to the end-to-end packet loss. To achieve better speech quality, several packet loss concealment techniques are applied before the recovery of the original real-time signals.

To provide a satisfactory level of QoS we have developed a "goal" function which aims at "Jitter" according to its definition in the real-time protocol RFC3393. We implemented new functions for CPN to support multiple QoS classes for multiple traffic flows, whereby different traffic flows (users) can declare their desired QoS goals before initiating a communication session at the same source node and then each flow is routed based on its own QoS criteria. Then, experiments were conducted to examine which QoS goal is better for RT packet delivery in a multiple traffic environment under varied traffic conditions via measurements of delay, delay variation (jitter) and loss.

Furthermore, we also consider packet end-to-end loss which consists of loss ocurring within the network, plus the packet discards happening inside the RTIP receiver which are affected by path switching induced by the adaptive nature of CPN. The correlation of packet end-to-end loss and path switching were carefully studied by an off-line analysis of packet traces from the CPN testbed network, together with a discrete event simulation of the behaviour of the re-sequencing buffer that resides in the RTIP receiver.

The main results of this paper are based on the measurements we conduct on the use of CPN to support RTIP. In particular, our work results in interesting insights which are counterintuitive:

- Our measurements compare the use of Delay and Jitter Minimisation as a means to offer QoS to RTIP connections. These measurements clearly show that using Jitter Minimisation as the QoS goal for routing all of the traffic flows in the network, namely the real-time traffic and the background (other) traffic, will not only minimise jitter but also delay and loss for real-time traffic.
- Furthermore we see that packet loss in the network is clearly correlated with path switching that is induced by the adaptive scheme that is inherent to CPN. However we also see that when path switching does not occur sufficiently often, peaks in packet loss can occur because *all the traffic* including the background non-RTIP traffic, will head for paths which are viewed to be good, resulting in unwanted congestion and hence loss, which in turn can only be mitigated by switching to less loaded paths and parts of the network.
- Finally, packet loss that can be provoked by path switching and congestion in a network, also results in further buffer overflows and hence further losses in the output re-sequencing buffers of the RTIP codecs which comes on top of the losses in the earlier stages of the network.

2 Real-Time Traffic over CPN for Multiple QoS Classes

As real-time needs are sensitive to the time-based QoS metrics "delay" on the one hand, and "delay variation" on the other, our experiments will consider both of these QoS classes. In fact we will allow the foreground or primary traffic class to have one or the other of these two QoS objectives, and similarly the background traffic will have one or the other of them as well. Thus our experiments on the CPN test-bed were conducted with real-time traffic and one of the two QoS requirements between arbitrary source-destination pair within the CPN testbed network, in the simultaneous presence of several background traffic flows with the same or the other QoS goal. The measurements we conducted were then used to see which of the QoS goals in effect provided better QoS for the real-time traffic and under which conditions of background traffic this was actually happening. Thus this section presents the implementation of the goal function for the QoS criterion of "Jitter", as well as the implementation of CPN that supports multiple QoS classes for multiple traffic flows simultaneously. We note that in previous experiments reported with CPN, all flows used the same QoS goal so that this change has required some significant modifications to the CPN software that is installed at each node.

2.1 Real-Time Packets over CPN with QoS Class Jitter

In RFC3393 and RFC5481, "Packet Delay Variation" is used to refer to "Jitter". One of the specific formulations of delay variation implemented in the industry is called Instantaneous packet delay variation ($IPDV$) which refers to the difference in packet delay between successive packets, where the reference

is the previous packet in the stream's sending sequence so that the reference changes for each packet in the stream.

The measurment of $IPDV$ for packets consecutively numbered $i = 1, 2, 3, \ldots$ is as follows. If S_i denotes the departure time of the i-th packet from the source node, and R_i denotes the arrival time of the i-th packet at the destination node, then the one-way delay of the i-th packet $D_i = R_i - S_i$, and $IPDV$ is

$$IPDV_i = |D_i - D_{i-1}| = |(R_i - S_i) - (R_{i-1} - S_{i-1})| \ . \tag{1}$$

To fulfill the QoS goal of minimising jitter, online measurement collects the jitter experienced by each DP. Since in CPN each DP carries the time stamp of its arrival instant at each node along its path, so when a DP say DP_i arrives at the destination, an ACK is generated with the arrival time-stamp provided by the DP, and as ACK_i heads back along the inverse path of the DP, and at each node the forward delay $Delay_i$ is estimated from this node to the destination by taking the difference between the current arrival time at the node and the time at which the DP_i reached the same node [32], divided by two. This quantity is deposited in the mailbox at the node. The instantaneous packet delay variation is computed as the difference between the value of $Delay_i$ and $Delay_{i-1}$ of the previous packet in the same traffic flow as in (1), and jitter is approximated by the smoothed exponential average of $IPDV$ with factor a smoothing factor 0.5:

$$\overline{J_i} = \frac{J_{i-1}}{2} + \frac{J_i}{2} \ . \tag{2}$$

Then, the corresponding value of jitter in the node's mailbox is also updated. When a subsequent SP for the QoS class of Jitter and the same destination enters the node, it uses data from the mailbox to compute the reward $Reward_i$ and then trigger the execution of the RNN which corresponds to the QoS class of Jitter and the destination to decide the outgoing link [32].

$$Reward_i = \frac{1}{\overline{J_i} + \epsilon} \tag{3}$$

where ϵ is used to ensure the denominator is non-zero.

2.2 Implementation of CPN Supporting Multiple QoS Classes

We enable CPN to support multiple QoS classes simultaneously, for multiple flows that originate at any node and each flow is routed based on its specific QoS criteria, we use the following components:

- The Traffic Differentiation can rely on source MAC (or IP), destination MAC (or IP) and the TCP/UDP port of the applications. For instance, the RTIP application "Linphone" has its dedicated SIP port (5060) and audio port (7080), which resides in the fields of the UDP header so that real-time traffic can be differentiated.

- The QoS Class Assignment can be defined according to the QoS requirements of different users or applications, and is stored in a configuration file which is loaded into the memory while CPN is being initiated.
- We can then treat each traffic flow according to its QoS requirement using multiple RNNs at each node, where each RNN corresponds to a QoS class and a source-destination pair. For instance, a real-time traffic flow with a specified QoS class (eg. forwarding delay or jitter) traveling across CPN corresponds to an RNN at each node along its selected path. ACKs coming back from the DPs of the real-time flow are used to collect the measurements of the specific QoS metric of the real-time flow itself and deposite the measurement results in the corresponding mailbox at each node on their way back to the source. SPs decide the outgoing link at each node they arrive by selecting the most excited neuron in the RNN based on the measurement result in the mailbox.

3 Measurement Methodology for Real-Time Packet Path Switching, Reordering and End-to-End Loss

CPN adaptively selects the path that provides best possible QoS requested for traffic transmission, leading to possible path switches so that traffic may suffer packet de-sequencing and loss. Various techniques for mitigating this effect have been demonstrated, such as the use of a switching probability that avoids simultaneous conflicting path switches among distinct flows that share some common routers, or the use of a minimum "improvement threshold" so that path switches occur only if the expected improvement in QoS is sufficiently large [28]. Accordingly, we are interested in examining the correlation between undesirable effects such as packet de-sequencing and end-to-end loss, and path switching. In the following sections, we described methods to carry out measurements and statistics for the three metrics. The measurements based on the off-line analysis of packet data which were captured by running "TCPDUMP" at the source and destination node so as to obtain the most detailed information of each packet.

3.1 Packet Path Switching

For a given flow, the path traversed by packets may change from time to time so as to satisfy the specific QoS requirement. This routing information provided by the corresponding SP is encapsulated into the cognitive map field of the DP while as it originates from the source node. Thus, the path used by each DP can be detected by extracting its routing information field after being captured at the source node. The metric we are interested in is the path switching ratio, which is defined as:

$$Ratio_{\text{path}} = \frac{Q_{\text{path}}}{N} \tag{4}$$

where Q_{path} is the number of path switches in a given flow during the time interval being considered, and N is the total number of packets forwarded in

that time interval. We can also define the path switching rate as:

$$Rate_{\text{path}} = \frac{Q_{\text{path}}}{T} \tag{5}$$

where T is the length of the time interval.

3.2 Packet Reordering

Packet reordering or re-sequencing is an important metric for real-time because packets have to be forwarded to the end user sequentially at the receiver in the same order that they have been sent, and those that arrive later than the required maximum will have to be discarded. When packets travel over multiple paths, packets sent later may actually arrive at the receiver before their predecessors, so that to obtain a fully correct playback, packets have to be stored until all their predecessors have arrived. However the reordering buffer at the receiver is necessarily of finite length, so that packets arriving to a buffer that is full will be discarded, and packets will have to be forwarded after a given time-out even when their predecessors have not arrived in order to avoid excessive time gaps with their predecessors that have already been played back. Packet reordering is done according to the recommendation from [20], which is based on the monotonic ordering of sequence numbers. Specifically, we add a 4-byte field in the CPN header to store the unique sequence identifier which is incremented by 1 each time a new packet is sent into a traffic flow. In addition, the sequence number residing in the Real-Time Protocol (RTP) header can also be used as the identifier, which is strictly monotonically increasing with increments of "320". To detect packet reordering, at the receiver we reproduce the sender's identifier function. The Next Expected Identifier is determined by the most recently received in-order packet plus the increment (denoted by Seq_{inc}) and denoted by $NextExp$. Thus for CPN the increment is "1", while for the RTP sequence number its "320". If S is the identifer of the currently arrived packet at the receiver, if $S < NextExp$, the packet is reordered and the number of reordered packest $Q_{\text{reordered}}$ is incremented by 1, else the packet is in-order and $NextExp$ is updated to $S + Seq_{\text{inc}}$. For example, if the packets arrive with the identifiers $1, 2, 5, 3, 4, 6$, *packet* 3 and *packet* 4 will be reordered.

To quantify the degree of de-sequencing, we also defined the "Packet Reordering Ratio/Rate" similar to (4) and (5), and the "Packet Reordering Density" denoted by $Density_r$, so that we may differentiate between isolated and bursty packet reordering as well as to measure the degree of burstiness of packet reordering, which may affect the packet drop rate of the re-sequencing buffer at the receiver. $Density_r$ is calculated as:

$$Density_r+ = \begin{cases} Cout_r^2 & \text{for bursty packet reordering} \\ Cout_r & \text{for isolated packet reordering} \end{cases} \tag{6}$$

where $Cout_r$ is the number of successively reordered packets; it resets to zero when the in-order packets arrive and is incremented when reordering occurs.

3.3 Packet End-to-End Loss

The recommendation in [19] states that packet loss should be reported "separately on packets lost in a network, and those that have been received but then discarded by the jitter buffer" at the receiver for real-time packet delivery, because both have an equal effect on the quality of real-time services. The combination of packet loss and packet discard is usually called packet end-to-end loss. In this section, we present the approaches we use to study these two metrics.

Here, packet loss will only refer to the packets lost in the network, as detected for a packet that is sent out but not received by its destination node. Since the sending packets are consecutively numbered by monotonically increasing identifiers at the source while entering CPN 3.2, for sent packets falling within the test interval, each packet received at the destination node will be matched with the combination of packet identifier, the source and destination IP address. Non-matched sent packets are identified as the lost packets, so that the timestamp of each lost packet and the number of lost pakcets are obtained. The packet loss Ratio/Rate is a commonly-used metric to quantify the degree of loss as in (4) and (5).

At the receiver in the RTIP application at CPN node, the received packets are stored in the "jitter buffer" which reorders packets and buffers them to reduce jitter. Packets that arrive later than the expected time which is required for the real-time signal recovery, and those that cause buffer overflow, will be discarded, contributing to the end-to-end loss. To achieve better speech quality, RTIP applications provide several packet loss concealment approaches to compensate packet loss before real-time playback. Thus, packet discard cannot be measured inside a RTIP application, and we cannot directly access the run-time version of the RTIP application. Accordingly, we have had to simulate the operation of a jitter buffer which employs resequecing so as to study packet discards and the buffer queue length, and their correlation with packet reordering and packet loss.

The discrete event simulation approach we use simulates the Arrival Events which correspond to the arrival of packets at the receiver, the Departure Events which indicate the departure of packets after being processed by the server, and the Wait Event. Due to the real-time demand of real-time traffic, the waiting time of packets that are delayed for re-sequencing in the buffer will have an upper bound. Each of this events are stored in the Future Event List (FEL) and are executed in time sequence. The queue used in the simulation consists of a waiting queue $LQ_{\text{reordered}}(t)$ for packets waiting for re-sequencing, and a processing queue $LQ(t)$, for packets waiting to be delivered to the end user. The event scheduling process which is integrated with the Re-sequencing Algorithm is as follows.

To capture a packet trace, we collect the packets in a real-time stream received at the CPN destination node during a test interval. The arrival of each packet corresponds to an arrival event scheduled in the FEL with its arrival time. It is assumed that the service time of the server follows an exponential distribution with average value 1 ms. We define $WT_{\text{threshold}}$ as the maximum

wait time for a packet and assign it the value of 200 ms which is substantially larger than the typical inter-arrival time of real-time packets is 20 ms. As has been mentioned in Sect. 3.2, $NextExp$, a key variable to identify the next expected packet consistent with the sending sequence for reordering detection, is initialised as the identifier of the first received in-order packet and updated by reproducing the sender's identifier function.

For each arrival event, If $Current\_Event\_ID > NextExp$, the packet should be delayed in the waiting queue (increament $LQ_{\text{reordered}}(t)$ by 1) to wait for the expected in-order packets. Moreover, if the early arrived packet waits for more than one packet, the sequence discontinuity size (indicated by d_s) is required to indicate the difference between the identifier of the current arrived packet and the $NextExp$. Therefore, the waiting time is calculated as:

$$tw = d_s * WT_{\text{threshold}} \ . \tag{7}$$

This produces a wait event which is scheduled to be executed at the time $t + tw$, where t is the arrival time of the current packet. The wait event executes when the waiting time of the packet in the waiting queue expires. The packet is then ordered at the head of the queue and the $NextExp$ is assigned the identifier of this packet and incremented by Seq_{inc} which is a constant equal to the increment between the identifiers of the two successive packets at the sender. Subsequently, the checking loop starts in the waiting queue. Each remaining packet in the waiting queue $LQ_{\text{reordered}}(t)$ will be checked to find the next expected packet. If the next expected packet is found, it is ordered and moved in the processing queue ($LQ = LQ + 1$) and $NextExp$ is udated as described above again. In addition, each time the expected packet arrives or is found in the waiting queue, the waiting time of each waiting packet is reduced by the length of the waiting threshold $tw \leftarrow tw - WT_{\text{threshold}}$. The checking runs until there none of the next expected packets are in the waiting queue.

If $Current\_Event\_ID == NextExp$, the packet will be buffered in the processing queue ($LQ \leftarrow LQ + 1$) if the server is busy; otherwise, the packet will be processed by the server, which produces the departure event with the occurrence time at $t + ts$ (ts is the simulated service time of the server). $NextExp$ is incremented by Seq_{inc}. Subsequently, the same checking loop executes in the waiting queue as described above to reorder the packets. On the other hand if $Current\_Event\_ID < NextExp$, the packet will be discarded because it arrives too late for real-time playback, and we increment $Q_{\text{discarded}}$ by 1.

During the simulation, we count the number of discarded packets $Q_{\text{discarded}}$ due to excessively late arrivals, as well as the time of the occurrence of the discard. The summation of $LQ(t)$ and $LQ_{\text{reordered}}(t)$ accounts for the total queue length in the re-sequencing buffer observed at time t and packet end-to-end loss is measured. To record the difference between isolated and bursty packet loss, we have defined the Packet End-to-End Loss Density and measure it similarly to $PacketReorderingDensity$.

4 Experimental Results

Our experiments were carried out on a wired test-bed network consisting of 8 nodes with the topology shown in Fig. 2, whereby multiple paths are available for packet delivery between source-destination pairs. CPN was installed as a loadable kernel module [33] running under Linux 2.6.32 at each node. Adjacent nodes are connected with 100 Mbps Ethernet links. The purpose of experiments is to examine two issues:

1. Network performance for real-time traffic with respect to the packet delay, jitter and packet loss, under varied background and obstructing traffic loads that use different QoS goals in their routing algorithm.
2. The correlation of packet end-to-end loss and path switching.

Note that we use 20 % of SPs in the total traffic and the real-time traffic rate is at 50 pps.

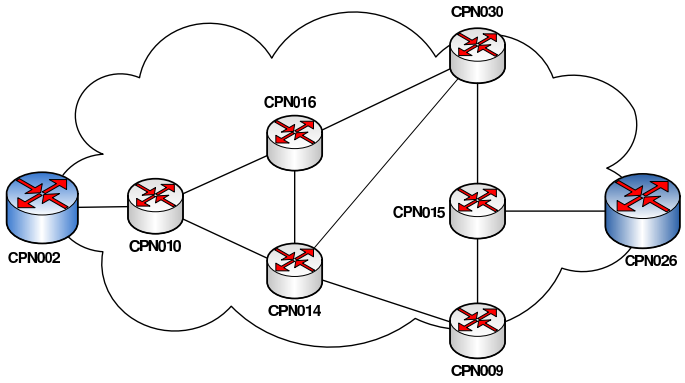

Fig. 2. CPN network testbed topology used in the experiments

To generate actual real-time traffic, "Linphone", a RTIP phone, was installed at each node in the network testbed, whereby real-time traffic can be originated everywhere within the network. Its SIP port is 5060 and audio port is 0780. Thus, real-time traffic can be differentiated according to the UDP port number exclusively used by Linphone. We used three flows of real-time traffic as shown in Table 1.

Due to the constant rate of the real-time traffic produced by "Linphone", background traffic flows with a range of data rates and constant packet sizes of 1024 bytes were introduced into CPN and allowed to travel between any source-destination pair to provide varied traffic conditions. For the sake of simplicity, we used six UDP traffic flows as the backgroud traffic, which were distributed as shown in the Table 2. We repeated each experiment with data rates of 1 Mbps, 2 Mbps, 3.2 Mbps, 6.4 Mbps, 10 Mbps, 15 Mbps, 20 Mbps, 25 Mbps, and 30 Mbps.

Table 1. Real-time Traffic Distribution used in the experiments

Source Node	Destination Node	Traffic Rate
cpn002	cpn026	50 pps
cpn010	cpn015	50 pps
cpn016	cpn009	50 pps

Table 2. Background Traffic Distribution used in the experiments

Source Node	Destination Node
cpn002	cpn026
cpn010	cpn015
cpn016	cpn009
cpn030	cpn010
cpn014	cpn002
cpn015	cpn016

Furthermore, CPN routing was implemented both for the real-time and the background traffic, and we set a distinct QoS goal setting for the real-time traffic and the background UDP traffic. To this effect, we used three scenarios as shown in Table 3. Thus in one case we allowed both the real-time and the background traffic to use Delay Minimisation as their QoS goal, in two other cases we used Jitter and Delay Minimisation (and vice-versa) for the two types of traffic, and finally we used Jitter Minimisation for both the real-time and the background traffic, and we report measurements for each of these cases.

Table 3. Distinct QoS goal combinations used in our measurements for the real-time and the background UDP traffic flows

Real-Time Traffic	Background Traffic
QoS:Delay	QoS:Delay
QoS:Jitter	QoS:Delay
QoS:Jitter	QoS:Jitter

For each test, three flows of real-time packets and six flows of background packets with a specified rate were originated using one of the three QoS goal setting scenarios for a duration of ten minutes. The real-time traffic flow from CPN002 to CPN026 was selected as the object of our measurement, since it had the greatest number of intermediate nodes between this source and destination pair in the testbed. The average delay and delay variation, and the packet loss ratio for this flow were measured so as to examine which QoS goal is better for real-time packets delivery in the multiple QoS goal environment we considered under widely varying traffic conditions.

From the results shown in the Figure 3, we can find the same trend for the three performance metrics in the three QoS goal setting scenarios under varying

traffic conditions. With low background traffic load, the two different QoS goals (Delay and Jitter) that are used for either the real-time or the background traffic have little effect, as may be expected, since overall delay, jitter and loss are very low.

We see quite clearly, that using Jitter as the QoS goal for *both* the real-time itself and the background traffic (BT) provides the *best results* for the real-time traffic at medium to high loads. Surprisingly enough, using Jitter for real-time and Delay for the background traffic (BT) yields the worst results. While, quite surprisingly, if Delay is used *both* for real-time and the background traffic, then the results in all three metrics (delay, jitter and loss) are not as good as when Jitter is used for both, but provides an intermediate result.

To explain this results, we have to refer to the definition of *Jitter* which means delay variation. Adaptive routing will necessarily cause some path switching, as well as some resulting end-to-end packet loss, and switching causes significant delay variation. Thus the QoS requirement of Jitter Minimisation applied to both real-time and the background traffic is likely to result in the least amount of path switching, even though at heavy traffic some route oscillations will still occur and will potentially degrade network performance. Therefore overall, we see that routing based on the QoS goal of *Jitter* is effective in alleviating route oscillations and losses, although the fact that it seems to reduce end-to-end delay itself (top figure) was definitely an unexpected result of these experiments.

5 Correlation Between Real-Time Packet Path Switching, Reordering and End-to-End Loss

While the previous result show that Jitter Minimisation is overall useful in the network to better satisfy the QoS needs of real-time traffic, not just when it is applied directly to real-time traffic but also when it is used for the background traffic, we were also interested in better understanding the detailed behaviour of the real-time traffic during these experiments. Thus we looked more carefully at the data collected in the experiments of Sect. 4 and measured the real-time traffic flow between CPN002 and CPN026 during the test interval being considered.

As summarised in Fig. 3, there is negligible packet loss under low traffic load. As we increase the background traffic rate, Fig. 4 shows us that as the six background traffic flows reach 20 Mbps, looking at the timestamp at which packets are send (the *x-axis*), the path switching rate of real-time traffic is around 10 packets per second. Though most of the time hardly any packets are lost, *we were indeed surprised to observe that in several time intervals there was was indeed a burst of packet loss.* During three time intervals (800–900 s, 900–1000 s, 1300–1400 s), *packet loss occurred in bursts while the path switching rates became very low.* This is the contrary of what we would have expected. However we feel that this does have an intuitive explanation.

The explanation is that when a given path for real-time traffic satisfies the Jitter Minimisation QoS criterion for a relatively long time, and hence the path switching rate is close to "0", *this path becomes attractive for background traffic*

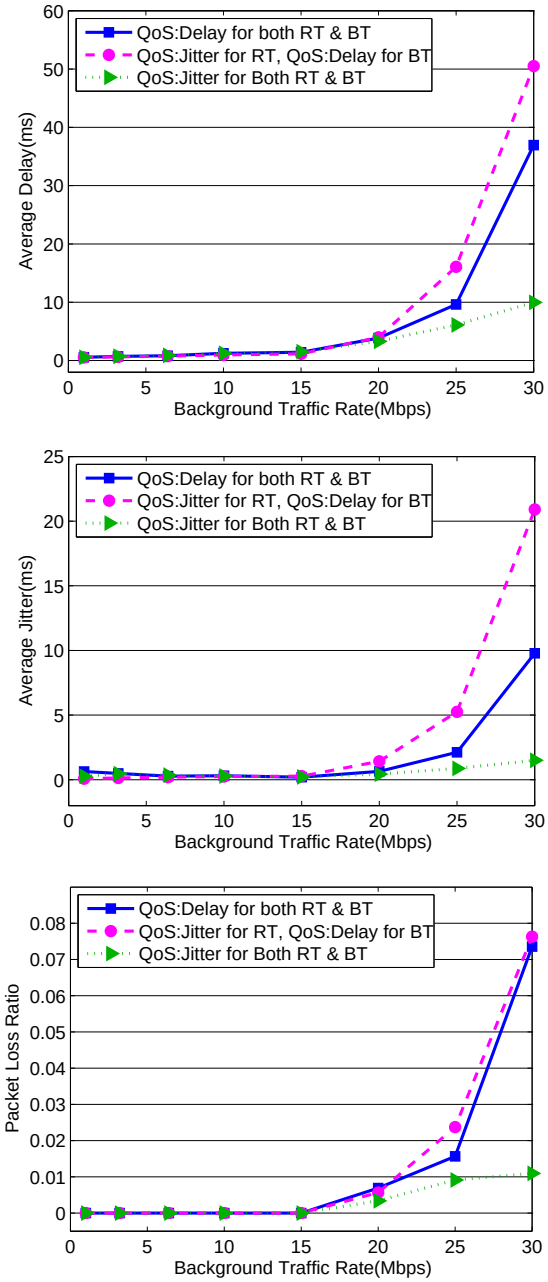

Fig. 3. The performance for real-time Traffic under varied background traffic conditions. We show the average delay (top), the delay variation or jitter (middle), and the packet loss ratio (bottom) for the real-time flow from CPN002 to CPN026 for different values of the background traffic load.

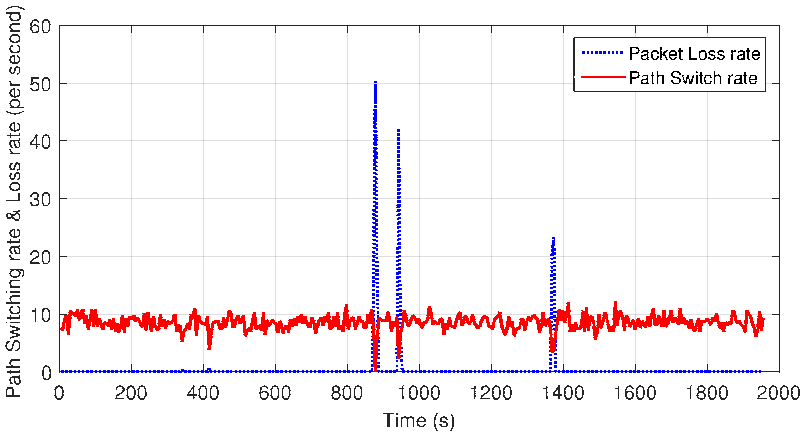

Fig. 4. The correlation of Packet Loss and Path Switching under medium traffic conditions: an apparently good path attracts more background traffic (BT), resulting in spurious bursty performance degradation and packet loss, followed by switching to better paths thanks to CPN's ability to adapt

and then becomes saturated, resulting in bursty packet loss with the loss rate increasing sharply and the loss ratio reaching "1". However, shortly after that, CPN reacts as it should to this QoS degradation: the SPs detect the performance degradation and another path is selected. Subsequently, loss rate decreases to "0" and the path switching rate increases.

As the rate of the six background traffic flows increased to 30 Mbps respectively, loss occurred more frequently and a large amount of packet de-sequencing was observed at the destination node. Figure 5 describes the the correlation of Packet End-to-End Loss, Reordering and Path Switching under heavy traffic conditions in a test run of duration of 700 seconds.

We can see that packet reordering rate (caused by de-sequencing) varies in proportion to path switching. This strong linear correlation between the two metrics is confirmed by regression analysis, showing that packet reordering in CPN is mainly due to path switching. Furthermore, it was found that packets were lost more frequently when the loss rates/ratio were not high.

It is not easy to observe the correlation of packet path switching and packet loss from the figure. By applying regression analysis, we also obtained a very weak correlation of the two metrics. It is possibly because under heavy traffic conditions, packet loss is not only due to link saturated, route oscillation induced by heavy traffic loads also leads to the occurrence of loss. We can found that the proper path swicthing rate (or ratio) is beneficial to loss reduction. If path switching rate is increased excessively, it is converted to route oscillation which also lead to packet loss.

We have observed that packet discards in the re-sequencing buffer contributes to packet end-to-end loss for real-time traffic, and heavy traffic load does provoke packet reordering and loss. To illustrate this, we used the real-time packets

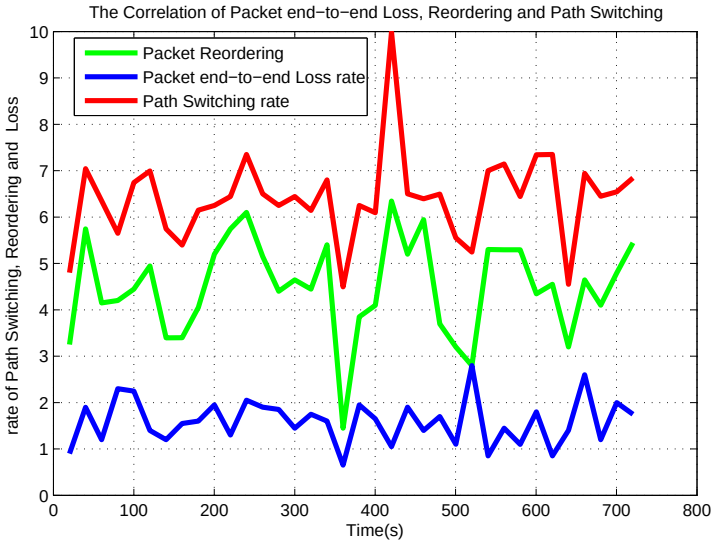

Fig. 5. The correlation of Packet end-to-end Loss, reordering and Path Switching under heavy traffic conditions: the data that is collected in the experiment, is processed using regression analysis

traveling between CPN002 at CPN026 in an experiment where six background traffic flows were simultaneously forwarded at a rate of 30 Mbps, under the same conditions as those discussed in the simulation of Sect. 3.3. It was found that the successive occurrence of bursty packet loss and reordering, are together the main reason for the significant increase in queue length at the re-sequencing buffer, causing buffer overflow. Note that packet loss in the network will cause packets to wait for their predecessors (who do not arrive) in the re-sequencing buffer; thus paradoxically buffer length increases when packets are lost. The higher the degree of burstiness of packet reordering and loss, the longer the waiting queue in the re-sequencing buffer and hence the higher the probability of buffer overflow.

Thus, the reordering density and loss density defined in Sect. 3.2 and 3.3, together with the queue length translated into the length of time spent in the buffer, are shown in Fig. 6. We can see that during the measurements queue length was relatively small when the reordering and loss density were low. At the time of around 520 seconds, many packets were lost or reordered successively, and subsequently the queue length in the buffer increased sharply. This data confirms our previous statement that packet reordering provokes large buffer queue lengths and actually accentuates (or creates a positive correlation) to the overall end-to-end packet loss, which results from the initial packet losses plus the buffer overflows.

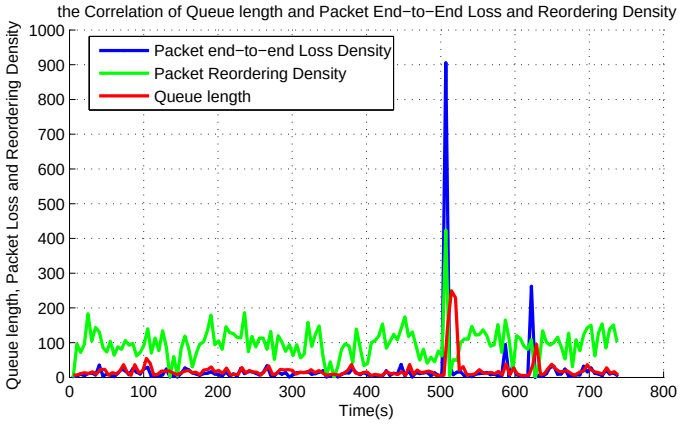

Fig. 6. The Correlation of the Queue Length in the re-sequencing Buffer, with Packet Loss and Reordering

6 Conclusions

This paper has presented an extension of the CPN routing algorithm so that multiple QoS classes can inhabit all routers of the network. Then these scheme has been applied to networks which carry real-time as well as other background traffic. Detailed experiments on CPN were conducted for real-time traffic, as well as other background traffic, with two distinct QoS requirements: *Delay* and *Jitter* Minimisation for any source-destination pair within the CPN testbed.

Surprisingly enough, measurement results for real-time traffic regarding average delay, jitter and loss have shown that when "Jitter" is specified as the QoS goal for both real-time and background traffic, better performance is achieved for all QoS metrics and all network load conditions. This seems to result from the useful effect of "Jitter" as a means to reduce route oscillations, also generally giving rise to less packet loss. However, we notice that long periods of path stability can also result in congestion with all traffic tending to use the same paths, when these paths do not oscillate and hence have low jitter: at that point, bursty effects of loss occur and CPN can mitigate for them by switching paths. Thus novel sensible routing schemes which distribute traffic over many paths in the network [34] may be useful even though they may result in the other disadvantages of multi-path packet forwarding. Furthermore we observe that packet loss in some parts of the network will provoke delays for other packets in the re-sequencing output RTIP codec buffers, which in turn can provoke buffer overflow and further losses. Thus, through a detailed and careful measurement study in a well instrumented network test-bed, this paper presents a complex series of cause and effects that allow us to better understand and better design networks that need to support real-time traffic. Future work in this area is planned to consider a specific application in distributed industrial control of manufacturing systems, so as to apply this research in a specific practical context.

References

1. FCC: 2013 Measuring Broadband America. In: Office of Engineering and Technology and Consumer and Governmental Affairs Bureau, Washington, DC, USA, Federal Communications Commission (Feb 2013)
2. Baldi, M., Martin, J.D., Masala, E., Vesco, A.: Quality-oriented video transmission with pipeline forwarding. Broadcasting, IEEE Transactions on 54(3), 542–556 (Sep 2008)
3. Gelenbe, E., Cao, Y.: Autonomous search for mines. European Journal of Operational Research 108(2), 319–333 (1998)
4. Gelenbe, E., Wu, F.J.: Large scale simulation for human evacuation and rescue. Computers and Mathematics with Applications 64(12), 3869–3880 (2012)
5. Soucek, S., Sauter, T.: Quality of service concerns in IP-based control systems. IEEE Transactions on Industrial Electronics 51(6), 1249–1258 (2004)
6. Soldatos, J., Vayias, E., Kormentzas, G.: On the building blocks of quality of service in heterogeneous IP networks. Commun. Surveys Tuts. 7(1), 69–88 (Jan 2005)
7. Dong, H., Hussain, F.K.: Focused crawling for automatic service discovery, annotation, and classification in industrial digital ecosystems. IEEE Transactions on Industrial Electronics 58(6), 2106–2116 (2011)
8. Dong, H., Hussain, F.K., Chang, E.: A service search engine for the industrial digital ecosystems. IEEE Transactions on Industrial Electronics 58(6), 2183–2196 (2011)
9. Santos, R., Pedreiras, P., Almeida, L.: Demonstrating an enhanced ethernet switch supporting video sensing with dynamic QoS. In: DCOSS, pp. 293–294 (2012)
10. Borzemski, L., Kaminska-Chuchmala, A.: Distributed web systems performance forecasting using turning bands method. IEEE Transactions on Industrial Informatics 9(1), 254–261 (2013)
11. Toral-Cruz, H., Argaez-Xool, J., Estrada-Vargas, L., Torres-Roman, D.: An introduction to VoIP: End-to-end elements and QoS parameters. In: InTech, pp. 79–94 (2011)
12. Roychoudhuri, L., Al-Shaer, E.S.: Real-time packet loss prediction based on end-to-end delay variation. Network and Service Management, IEEE Transactions on 2(1), 29–38 (Nov 2005)
13. Canovas, S.R.M., Cugnasca, C.E.: Implementation of a control loop experiment in a network-based control system with lonworks technology and IP networks. IEEE Transactions on Industrial Electronics 57(11), 3857–3867 (2010)
14. Cucinotta, T., Mancina, A., Anastasi, G., Lipari, G., Mangeruca, L., Checcozzo, R., Rusina, F.: A real-time service-oriented architecture for industrial automation. IEEE Transactions on Industrial Informatics 5(3), 267–277 (2009)
15. Felser, M., Jasperneite, J., Gaj, P.: Guest editorial special section on distributed computer systems in industry. IEEE Transactions Industrial Informatics 9(1), 181 (2013)
16. Gaj, P., Jasperneite, J., Felser, M.: Computer communication within industrial distributed environment – a survey. IEEE Trans. Industrial Informatics 9(1), 182–189 (2013)
17. Gelenbe, E., Hussain, K., Kaptan, V.: Simulating autonomous agents in augmented reality. Journal of Systems and Software 74(3), 255–268 (2005)
18. Silvestre-Blanes, J., Almeida, L., Marau, R., Pedreiras, P.: Online qos management for multimedia real-time transmission in industrial networks. IEEE Transactions on Industrial Electronics 58(3), 1061–1071 (2011)

19. Friedman, T., Caceres, R., Clark, A.: RFC3611: RTP Control Protocol Extended Reports (RTCP XR). Request for Comments (Nov 2003)
20. Morton, A., Ciavattone, L., Ramachandran, G., Perser, J.: RFC4737: Packet reordering metrics. Request for Comments (Nov 2006)
21. Larzon, L.A., Degermark, M., Pink, S.: Requirements on the TCP/IP protocol stack for real-time communication in wireless environments. In: Quality of Service in Multiservice IP Networks. LNCS, vol. 1989, pp. 273–283. Springer (2001)
22. Gelenbe, E., Lent, R., Xu, Z.: Design and performance of cognitive packet networks. Performance Evaluation 46(2,3), 155–176 (Oct 2001)
23. Gelenbe, E.: Cognitive packet network. In: U.S. Patent 6,804,201. (Oct 2004)
24. Demichelis, C., Chimento, P.: IP packet delay variation metric for IP Performance Metrics (IPPM). In: https://www.ietf.org/rfc/rfc3393.txt, The Internet Society (Nov 2002)
25. Gelenbe, E., Kazhmaganbetova, Z.: Cognitive packet network for bilateral asymmetric connections. IEEE Transactions on Industrial Informatics 10(3), 1717–1725 (2014)
26. Gelenbe, E.: The first decade of g-networks. Europ. J. Operational Res. 126(2), 231–232 (2000)
27. Gelenbe, E., Lent, R., Nunez, A.: Self-aware networks and QoS. Proceedings of the IEEE 92(9), 1478–1489 (Sep 2004)
28. Gellman, M.: QoS Routing for Real-time Traffic. PhD thesis, Imperial College London (2007)
29. Baccelli, F., Gelenbe, E., Plateau, B.: An end-to-end approach to the resequencing problem. J. ACM 31(3), 474–485 (Jun 1984)
30. Masala, E., Quaglia, D., de Martin, J.C.: Variable time scale multimedia streaming over IP networks. Multimedia, IEEE Transactions on 10(8), 1657–1670 (Dec 2008)
31. Baldi, M., Marchetto, G.: Time-driven priority router implementation: Analysis and experiments. Computers, IEEE Transactions on 62(5), 1017–1030 (May 2013)
32. Gelenbe, E., Lent, R., Montuori, A., Xu, Z.: Cognitive packet networks: QoS and performance. In: Proceedings of the IEEE MASCOTS Conference, Ft. Worth, TX, pp. 3–12. Opening Keynote Paper (Octr 2002)
33. Gelenbe, E.: Steps toward self-aware networks. Comm. ACM 52(7), 66–75 (Jul 2009)
34. Gelenbe, E.: Sensible decisions based on QoS. Computational Management Science 1(1), 1–14 (2003)