Smart Routing in Wired and Wireless Networks

by
Zarina Kazhmaganbetova

Supervisor: Professor E. Gelenbe

A Thesis submitted in fulfilment of requirements for the degree of
Master of Science and Diploma of Imperial College
Communications and Signal Processing
of Imperial College London

Department of Electrical and Electronic Engineering
Imperial College London
August 27, 2013

Abstract

This project is based on the previous research in Cognitive Packet Networks, but introduces new approach for QoS-based bilateral traffic differentiation. CPN protocol benefits from the algorithms of user-defined QoS routing. Any user is independent to choose a QoS that would suit its requirements to the path depending on type of data sent. The QoS is defined beforehand and does not change for a given user traffic transmission. This project innovates by introducing QoS differentiation between user's transmission directions. The idea is that in two users bilateral communication scheme, each user's node turns out to be a source and a destination at the same time, managing two types of flows. Traffic, originated by a user, is regarded as Uplink, and traffic, sent back by a user in response, is regarded as Downlink. In fact, demandining four distinct QoSes per direction, this project offers another approach that is differentating QoS between sender node roles (source or destination) instead of user's traffic direction (uplink and downlink). Traffic volume assymetry between received and sent data was taken as a triggering condition. According to this the smaller traffic is ruled by less Delay QoS and the larger traffic - by less DP Loss QoS. DP Loss QoS was implemented in the packets originating nodes and was additionally improved with multiple path tracking algorithm. Owing to several path availability and their tracking, a sender can always select the less DP Loss path for transmission.

Acknowledgment

I would like to express my gratitude to Professor Erol Gelenbe, my Project Supervisor. Without his advise and explanations this MSc thesis could have not been finished. I was inspired by Prof. Gelenbe ideas while developing the main objective of the project. I admire the level of Prof. Gelenbe expertise, and my project highly benefited from his constructive critisism. It was an outstanding and exciting experience to work under Prof. Gelenbe supervision.

I highly appreciate the help and guidance provided by Dr Ricardo Lent for the programming part ot the project. Receiving explanations from the programmer of the first version of Cognitive Packet Network protocol was extremely useful for my ideas implementation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Cognitive Packet Network (CPN) is a Software Enabled Network that manages an arbitrary number of network users over an open source LINUX based infrastructure, and offers differentiated quality of service (QoS) over different network paths [1], [2], [3]. 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. Thus a user selects the best path among the choices it has been offered so that its traffic can reach the destination based on the user's decision in the framework of its own QoS objective or Goal. The QoS objectives for a user in CPN can be the conventional metrics that are typically optimized in networks, such as end-to-end packet delay, loss or jitter, or a combination thereof [4]. However CPN has also been used to achieve the optimization of more sophisticated QoS Goals such as security [5], [6] and energy savings [7], [8], or a combination of energy savings and delay [9]. This research innovates with respect to previous work on CPN by addressing QoS Goals that may be asymmetric with respect to traffic that is being forwarded from a source to a destination, and with respect to the traffic that the source receives from the destination. Thus in this paper we investigate the uplink and downlink traffic differentiation using the Cognitive Packet Network (CPN) paradigm. Uplink traffic is defined as data that travels from the source towards destination. The initial source is regarded as the Uplink Sender (US).

Downlink traffic is data that travels back from the destination towards the source. The destination server is then also a sender, but is distinguished as a Downlink Sender (DS). The overall scheme we consider is shown in Figure 1.1. Most of the applications that may use this described scheme of communication can require different QoS for either uplink or downlink transmission. If the uplink traffic needs the shortest packet delay, it may not be the case for the downlink traffic. For the downlink branch it could be more important to have a more secure path, or less loss, instead of shortest delay.

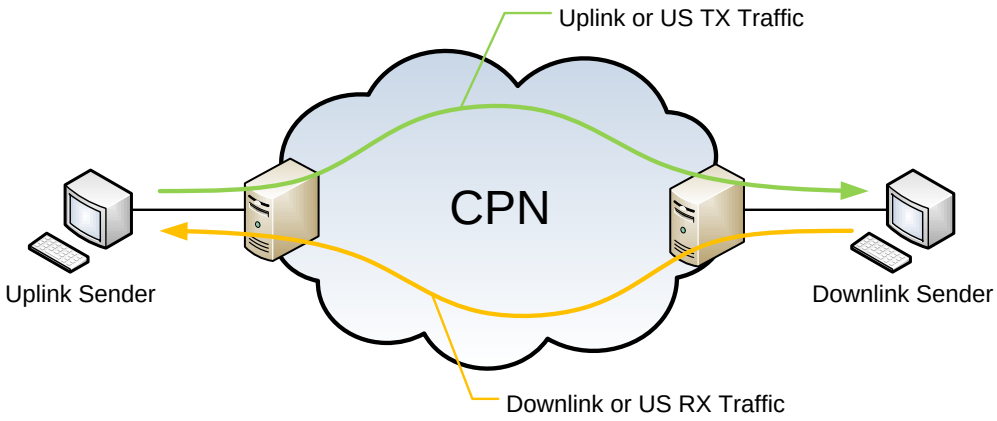

Figure 1.1: The CPN network model for traffic differentiation per node.

There are different approaches that one may use to differentiate between the uplink and downlink QoS Goal. One can obviously treat them as two distinct connections with the Source of one being the Destination of the other, and vice-versa. If the source and destination play a symmetric role, then both of them will have an uplink and downlink behaviour, so that both of these two "connections" may be differentiating in terms of the QoS Goal with respect to the data they are sending. For instance the node A (in an A to B and vice-versa connection) will sometimes be sending data to B as an Uplink, and sometimes as a Downlink that is going back from A to B. Thus the traffic flows from A to B may have differentiated QoS, and hence also distinct paths, based on whether the traffic from A to B is an Uplink or Downlink traffic (see Figure 1.2).

One can of course, create four distinct connections, two from A to B (Uplink and Downlink) and similarly from B to A, with distinct QoS Goals, and possibly distinct paths being utilized. However a simpler scheme would use a trigger so that based on the role

it is assuming, the source can become a destination, and vice-versa, so that it can switch its role, and hence also A and B can change the QoS Goal that they use based on the data that they are forwarding. This research will investigate the latter approach because it offers a uniform way of handling such bilateral and potentially asymmetric connections.

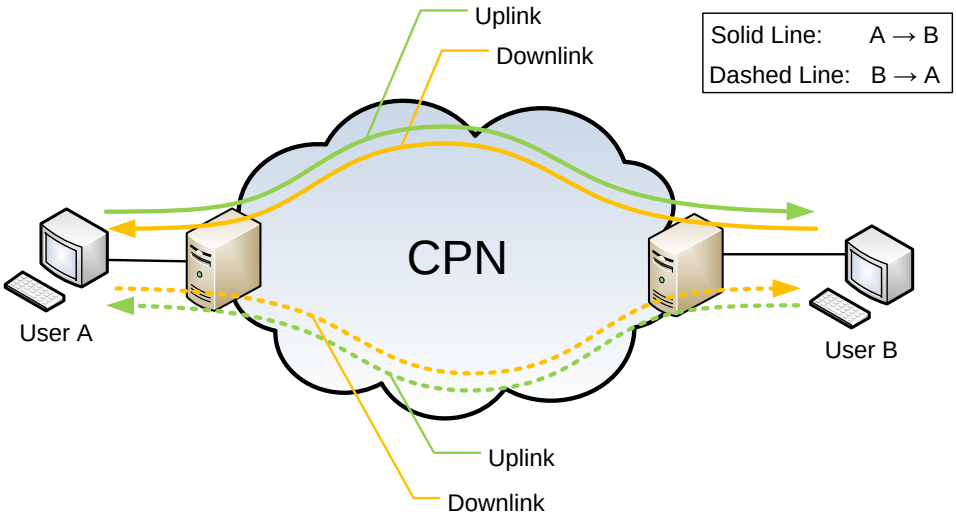

Figure 1.2: The CPN network model for traffic differentiation per user.

1.1 The Cognitive Packet Network (CPN)

The Cognitive Packet Network is a Self-Aware network that uses adaptive algorithms for path searching without routing tables, available in the nodes. The absence of routing tables means that every node in CPN is aware about its immediate neighbours only. Thus, the paths are searched on demand, when there is a payload packet for transmission. This property distinguishes CPN from other types of Self-Aware Networks. CPN is not loaded with packets flows that are generated for keeping routing tables up-to-date. CPN possesses the QoS driven search algorithm [10]. The variety of modern applications that use networks could not be satisfied with one shortest path QoS. Some applications could demand other QoS (loss, security, delay) accepting the longer path. CPN provides algorithms that search, maintain and update path depending on the distinct QoS Identifier stamped on the CPN packets.

CPN is a software based network and is implemented on Linux infrastructure. CPN is integrated with IP protocol, IP packets are treated as a payload [11]. IP packets are encapsulated into CPN packets with CPN header added, after that CPN packets are passed by the code to Data Link Protocol Layer. Data Link Protocol depends on the physical media used for transmission. In this project Ethernet protocol was used. This section gives detail description of CPN protocol elements, algorithms used for adaptive routing, such as Random Neuron Network(RNN) and Reinforcement Learning (RL).

1.1.1 CPN Elements

There are three distinct types of packets in the CPN. CPN packets have the same format but different roles. These packets are: **Smart Packet** (SP), **Dumb Packet** (DP) and **Acknowledgement Packet** (ACK). ACK packets are also divided into two groups: Smart Packet Acknowledgement (SACK) and Dumb Packet Acknowledgement (DACK).

The CPN packet format is given in the Figure 1.3. The CPN packet has a variable length that mainly depends on the route length. The packet consists of such fields as: CPN Header, Route field, Cognitive Map (CM) and Data field.

The **CPN Header** has several sub-fields:

- **Version**. This field contains CPN version number. Current is 3.
- **Pkt Type**. This field defines the type of the CPN packet: SP, SACK, DP or DACK.
- **Proto**. This field contains the QoS Identifier that is used for a given packet.
- **Route Length**. The length of the route in number of hops excluding source and destination is written into the field.
- **Flags**. The content of the field depends on the QoS stamped on the packet. This field defines the CM structure.
- **CM Index**. This is a pointer field that stores the index number of the current node in the whole route. Respectively, CM Index is zero for the source node, and CM Index is $(CMRouteLength + 1)$ for the destination node.
- **User ID**. This field was not in original CPN design and was added for this project

fulfilment. The field stores a user's numeric identifier that originates DPs. DACKs that travel back in response to DPs has the same User ID.

- **Route ID**. This field was not in original CPN design and was added for this project fulfilment. This field contains the Path ID that has been selected for the DP travelling.
- **Destination Address**. The IP address of the destination node is written into this field.
- **Source Address**. The IP address of the source node is written into this field.

Route field stores the ordered sequence of intermediate nodes from the source towards the destination.

CM field stores the QoS measurement used for RL. Depending on the QoS algorithm CM could store a vector of values, one value or could be empty. If CM is a vector, each vector element stores distinct measurement value per every node in the route. If CM is a field, it is recurrently updated while a packet travels through the route. If CM is empty, no measurements are collected for a given QoS.

Data field is designed for payload.

Each type of packet in CPN was designed to fulfil unique functions that are described below.

SP is used for adaptive routing, this packet searches for the path towards the destination. There are two available algorithms for SP path searching: random search and Reinforcement Learning (RL). In the random search, after arriving to the node, SP selects the next hop randomly. For the RL next hop decision making, SP uses Random Neuron Networks (RNN) that reside at every node it visits. While travelling SP collects the QoS measurements and stores them into the CM. As CPN uses probabilistic approach for route discovering, SPs could not be allowed to search eternally. After a timeout a SP, that has not managed to reach the destination, is destroyed. SP does not carry any payload, so that the Data field of SP is Null.

SACK is a packet that is created upon SP arrival to the destination and its purpose is to bring back the discovered route to the source. SACK travels back to the source following the reverse route excluding the loops. During its travelling SACK updates the

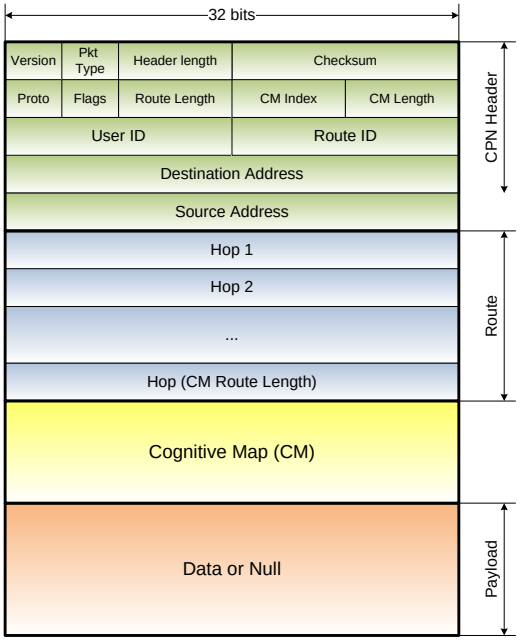

Figure 1.3: CPN packet format.

Mailboxes (MB), using the measurements collected by the original SP. MB values are used for RNN Rewards computing and weights update. SACK's Data field is also Null.

DP is a payload packet, it carries the IP packet that resides in the Data field of the packet. DP uses the route brought by the SACK to travel towards the destination. Upon arrival to every intermediate node DP does not interact with RNN or utilize a RL algorithm for next hop deciding. The whole route is recorded into the Route field of the packet, and DP follows it. For some QoS algorithms, DPs also collect the measurements during the travelling and store them into the CM.

DACK is generated in the destination node upon DP arrival. It travels back to the source, informing it that DP was delivered. For some QoS algorithms, DACK updates the MBs as SACK does.

One or several MBs reside in every CPN node. A distinct MB is created per Source, Destination, QoS Identifier combination. Each entry in MB refers to the Neighbour-Neuron Address. When SACK or DACK travels through the node, the corresponding Neuron entry value or MB value itself is updated with QoS measurement that was collected.

To conclude, CPN consists of the following elements: packets (SP, DP and ACK), RNNs and MBs. Packets are generated by the source and destination nodes. RNNs and

MBs are created and maintained in the intermediate nodes of CPN.

1.2 Random Neuron Network (RNN)

Random Neuron Networks implement the biological mechanism of interaction in computer networks. This type of recurrent networks was invented and proposed by E. Gelenbe [12], [13], [14]. In the RNN, consisting of n neurons, the communication between neurons is modelled as exchanging of positive (or excitatory) and negative (or inhibitory) signals. Additionally, neurons receive signals outside the network (exogenous signals), and signals can also leave network. The results of signals exchange is changing neurons' potential level, which is defined as a non-negative integer. Each positive signal increases neuron's potential by one, and negative decreases by one, so that these signals are the spikes of unit amplitude. If neuron potential is zero, before negative signal arrival, then it has no effect on the potential. Whenever neuron's potential is more than zero, it is regarded to be in the "excitation" state, so that it starts to fire or to generate spikes. On other hand, firing makes neuron to produce signals that also decreases this neuron's potential. Neuron i sends signal to neuron j with rate $r(i)$, so that intervals between firings are exponentially, independently and identically distributed. Signal could be a positive signal with probability $p^+(i,j)$, or inhibitory signal with probability $p^-(i,j)$, or it could leave the network with probability $d(i)$. Thus, the total transition probability for neuron i is equal 1:

$$d(i) + \sum_j [p^+(i,j) + p^-(i,j)] = 1, \qquad 1 \le i \le n \tag{1.1}$$

Neuron produces excitatory signals with rate $w^+(i,j) = p^+(i,j)r(i)$ and inhibitory signals with rate $w^+(i,j) = p^+(i,j)r(i)$. At the same time, neuron i receives exogenous signals that are modelled as stationary Poison processes of rate $\Lambda(i)$ for positive signals, and $\lambda(i)$ for negative.

At each moment of time t the RNN is characterized by vector of neurons' potentials $k(t) = (k_1(t), ..., k_n(t))$, where $k = (k_1, ..., k_n)$ is one distinct value of vector. Obviously, from the RNN model given above, $\{k(t), t \ge 0\}$ is continuous time Markov chain. If

stationary probability distribution exists $p(k) = \lim_{t\to\infty} Prob[k(t) = k]$, then it could be calculated by solving Chapman-Kolmogorov equations in steady-state given in 1.2:

$$\begin{aligned} & p(k)\sum_i [\Lambda(i) + (\lambda(i) + r(i))\mathtt{1}[k_i > 0]] = \\ & = \sum_i [p(k_i^+)r(i)d(i) + p(k_i^-)\Lambda(i)\mathtt{1}[k_i > 0] \\ & + p(k_i^+)\lambda(i) + \sum_j [p(k_{ij}^{+-})r(i)p^+(i,j)\mathtt{1}[k_j > 0] \\ & + p(k_{ij}^{++})r(i)p^-(i,j) + p(k_i^+)r(i)p^-(i,j)\mathtt{1}[k_j = 0]]] \end{aligned} \tag{1.2}$$

where `1[.]` is a function that is equal zero, if condition in the brackets is false, and equal to 1, if condition is true. The potential vector values are defined, as follows:

$$\begin{aligned} k_i^+ &= (k_1, ..., k_i + 1, k_j, ..., k_n) \\ k_i^- &= (k_1, ..., k_i, k_j + 1, ..., k_n) \\ k_{ij}^{+-} &= (k_1, ..., k_i + 1, k_j - 1, ..., k_n) \\ k_{ij}^{++} &= (k_1, ..., k_i + 1, k_j + 1, ..., k_n) \end{aligned} \tag{1.3}$$

These equations are also defined in the RNN theory as network global balance equations and they describe all the processes that occur in the network. It will be shown further that RNN possesses a graceful and computationally efficient solution for the stationary distribution calculation.

In the CPN design, RNN is used to perform routing decisions. One or several RNNs are created in every intermediate CPN node per Source, Destination and QoS Identifier Combination. Each neuron in RNN represents one immediate neighbour of a given CPN node. The RNN neuron properties apply to the CPN protocol, so that at the firing state neuron i produces excitatory or inhibitory signals with rates $w^+(i,j)$ and $w^-(i,j)$, $j = 1...n$. The CPN protocol benefits from RNN usage, because each CPN node requires information only about immediate neighbours to be able to perform routing decisions. When a next hop is to be selected for a packet with a corresponding Source, Destination

and QoS ID set, the algorithms tries to find the most "excited" neuron to send the packet to. For that purpose the steady-state and stationary distribution of the corresponding RNN should be found, if it exists. To solve the problem the product form solution is used.

The product form solution theorem is given below and its proof could be found in [12], [13], [14]. Let the stationary probability that neuron i is in firing state to be defined as:

$$q_i = \lim_{t\to\infty} Prob[q_i > 0] = \frac{\lambda^+(i)}{r(i) + \lambda^-(i)} \tag{1.4}$$

where $\lambda^+(i)$ and $\lambda^-(i)$ for $i = 1...n$ are defined by the system of non-linear equations, called signal flow equations:

$$\begin{aligned} \lambda^+(i) &= \sum_j q_j r(j) p^+(j,i) + \Lambda(i) = \sum_j q_j w^+(j,i) + \Lambda(i) \\ \lambda^-(i) &= \sum_j q_j r(j) p^-(j,i) + \lambda(i) = \sum_j q_j w^-(j,i) + \lambda(i) \end{aligned} \tag{1.5}$$

If a non-negative and unique solution exists for equations 1.5, such that $q_i < 1$, $1 \le i \le n$ then

$$p(k) = \lim_{t\to\infty} Prob[k(t) = k] = \prod_{i=1}^{n} (1 - q_i) q_i^{k_i} \tag{1.6}$$

The solution was proved to be unique [13]. The average potential of neuron i in steady state could be found as $A_i = q_i/(1-q_i)$, so that a neuron with the largest firing probability will have a largest potential. Thus, knowing the stationary probability of each neuron to be excited allows to estimate the neuron with the largest potential that is necessary for routing decision in CPN.

The RNN theory is not limited to model and solution given above. It was extended, so that new types of random networks were invented and investigated, such Gelenbe or Generalized Neuron Networks (GNN) and Multiple Signal Class Random Neuron Networks (MCRNN) [15], [16], [17], [18]. These types of networks model more generalized and complicated cases that could occur in real networks or applications.

In the GNN networks, the excitatory and inhibitory signals are accompanied by extended signals with a more sophisticated behaviour, like interactions [15], or neurons potential values vary in extended way, allowing bipolar values (BGNN), or clamping

(CGNN) [17], [18]. Synchronous interactions occur in the RNN when the firing in one neuron i results in a excitatory signal that arrives to another neuron j and triggers firing in it, so that the described cycle is repeated by the second neuron j over the third m or several neurons, producing a cascade of firings. Such extended behaviour introduces additional term into the global balance and signal flow non-linear equations $Q(i, j, m)$ and additional value of potential vector k, that describes the extended behaviour. The RNNs with interactions were investigated in the scope of clustering in large networks, when a groups of neurons closely interact, so that the changes in one trigger the same changes in neurons of one cluster. In BGNN all neurons are divided into two groups with positive and negative behaviour. Positive neurons exchange signals as ordinary RNN neurons. The negative neurons do it completely in the opposite way. Inhibitory signals increases the negative potential of the neuron, excitatory signals decreases it. If positive signal arrive to the negative neuron with zero potential it does not affect it. BGNN introduces changes to signal flow non-linear equations, so that rates for negative and positive neurons are treated separately. In CGNN the neuron's average potential is not allowed to be below some pre-defined constant, called a "clamping" constant. The both BGNN and CGNN are widely used as universal function approximators.

In MCRNN networks, there are several classes of excitatory and inhibitory signals that circulate in the network. Signals of one class change neuron potential of corresponding class, so that the resulting neuron potential is defined as a sum of all classes potentials. In particular, having a set of C classes in the RNN, consisting of n neurons, the potential of neuron i will be equal $k_i = \sum_{c=1}^{C} k_{ci}$. Each element in the potential vector is a vector itself, representing the values of class potentials in one neuron. The classes interact between each other, so that one class signal, triggered by the firing of the same class in one neuron, could cause the changes of another neuron potential in the same or another class. MCRNNs suit the requirements of complex and sophisticated applications that demand from RNN to process inputs of number of variables.

If steady-state exists, the stationary distribution for extended types of RNNs could be always found by means of product-form solution. The product-form solution is generalized for any extended type of RNN network, so that the stationary distribution is a

product of probabilities of neurons to be in excitation state. The main problem is the evaluating of the stationary distribution existence. The enough condition for any RNN, that helps to define existence, was provided in [19]. It advises to find the fixed point y^* (single class) or $\underline{y}^*$ (multi-class) for the given signal flow equations 1.5, which is always available according to Brouwer's theorem. Eventually, the stationary distribution $p(k) > 0$ exists, if $0 \leq q_i(y^*) \leq 1$, for $i = 1, ..., n$, and it is not available, if $q_i(y^*) > 1$. In other words, the product-form solution for extended RNNs is generalised as a fixed-point solution.

There is another remarkable property of RNN that makes it suitable for a vast of applications. This property is Learning [14], [19]. The RNN stationary probability distribution is updated through excitatory and inhibitory rates updating. In the RNN Learning theory these rates are also defined as weights. There are two types of Learning in RNN: Gradient-Descent Learning and Reinforcement Learning. The first type of RNN learning performs calculation of weight matrices W_k^+ and W_k^-, with purpose to provide the desired output with respect to given input. Consider the set of K input exogenous rates (excitatory Λ_k and inhibitory λ_k) and the set of desired and pre-defined neuron values $y_k = (y_{1k}, ..., y_{nk})$, the algorithms aims to minimize quadratic error or "cost" function:

$$E_k = \frac{1}{2} \sum_{i=1}^{n} a_i (q_i - y_{ik})^2, \qquad a_i > 0 \tag{1.7}$$

The stationary excitatory probabilities q_i are calculated with respect to input exogenous rates from the signal flow non-linear equations. The equation 1.4 can be re-written as follows:

$$q_i = \frac{N(i)}{D(i)} \tag{1.8}$$

where $N(i)$ and $D(i)$ are defined as

$$N(i) = \sum_j q_j w^+(j,i) + \Lambda(i)$$

$$D(i) = \sum_j q_j w^-(j,i) + \lambda(i) + r(i)$$

The weight update is conducted by formula, as follows:

$$w_k(u,v) = w_{k-1}(u,v) - \eta \sum_{i=1}^{n} a_i(q_i - y_{ik})[dq_i/dw(u,v)]_k \tag{1.9}$$

The term $w_k(u,v)$ is taken either equal to positive weights, or to negative weights. The η constant is a learning rate that is chosen arbitarly and could be updated during algorithm running. After the derivative values calculation the weight matrices are computed:

$$\begin{aligned} dq_i/dw^+(u,v) &= \gamma^+(u,v)q_u(I-W)^{-1} \\ dq_i/dw^-(u,v) &= \gamma^-(u,v)q_u(I-W)^{-1} \end{aligned} \tag{1.10}$$

In the equation 1.10 the value of W is equal to $[w^+(i,j) - w^-(i,j)]/D(i), i,j = 1,...,n$. The $\gamma^+(u,v)$ and $\gamma^-(u,v)$ term calculation and the Gradient-Descent algorithm steps could be found in [19]. The iterations for one successive value of input k are repeated, until the difference between to successive matrices is below some pre-defined threshold, or sometimes for calculation speed only one iteration is performed. The algorithm theory was extended to multi-class RNNs that sufficiently enlarged the applications of this learning technique [14].

The Reinforcement Learning differs from Gradient-Descent Learning and has an aim to update neurons values, to reward or to punish them, according to the goal function that is desired to be approached, and it is widely used in the network applications. The RL algorithm also updates the stationary distribution through weights update. This type of Learning is applied in the CPN protocol and described in details in the next section of this chapter.

As it was mentioned before, learning in RNN allows to perform functions approximation. After processing a set of input parameters RNN can approximate the relationship between incoming and output values in the form of some function. It means that RNN has been learnt. After that RNN will recognise learnt patterns and provide the desired output to any input of the same type. This property was utilised in the image recognition [20], and effective video compression [21]. Gradient-Descent Learning is used to train MCRNN to distinguish tumours or abnormal areas in Magnetic Resonance Images.

MCRNN divides image into regions and evaluates each region by trying to match them into known patterns. The unrecognised areas are named at the output by MCRNN as regions with abnormal state. Image and video processing is also important for real-time streaming network applications. The video compression benefits from RNNs, because neuron network can recognise the movement in the flow of successive snaps or images of the video stream. In fact, there are two RNNs used for this application, one is called Encoder RNN and another Decoder RNN. Encoder RNN processes the input blocks from larger number of neurons into the output blocks that are produced by the less number of neurons. Thus, the compression is conducted. Moreover, Encoder RNN is capable to recognise the static blocks of the video stream and decide not to process them and not to send over the transmission link. The quality of compressed video is increased, because more network bandwidth and speed is available for useful blocks transmission, as it is not loaded with redundant information carrying.

The applications of RNNs is not limited with image and video processing. This random networks were successfully implemented for solving optimization tasks. Optimization problems target is always to find the optimum solution for arranging a given set of entities in the way, so that the "cost" function is minimised. The RNN showed its consistency for heuristic algorithms of tasks assignments problem [22], and for graph optimization problems, like multicast routing [23] and travelling salesman problem [24]. Each neuron of RNN, implemented for task allocation, represents a task-processor pair, so that the most excited neuron will be selected imposing the corresponding processor to pick up the task. The large excitatory weights will exist on the links between frequently communicating processors. The constraint of the task is the maintaining the right order of the task flow, that will be controlled by the excitatory and inhibitory weights changing. The graph optimization tasks are performed in the form of iterative algorithm that runs on RNN, until convergence. From iteration to iteration the graph edges will endure inhibitory weights increase if they do not satisfy to the constraint or have already been picked up for the solution. The algorithm finishes, when the optimum solution is found.

From optimization tasks description there is a straightforward transition to RNNs application in networks. RNNs are used to make a routing decisions that suit optimally the

pre-defined QoS Goal function [10], [25], or to allocate resources in Admission Control (AC) algorithm [2], or to block malicious packets flows when DDoS attack happens [5]. For all network applications RNNs reside inside network nodes, so that each neuron corresponds to the immediate neighbour of the node (server, router). For routing decision making the network quality metrics, collected by packets, are used to learn RNN, so that a neuron with most promising direction for goal function is selected. As it was mentioned before this way RNNs are utilized in CPN protocol [25]. The CPN network protocol is also implemented for managing AC and DDoS. In AC application of RNN the congestion in the network is prevented, as a user will not be allowed to send packet over the network, if there is no path available to it, which satisfies the desired QoS function. If network saturation is increasing, the number of paths, satisfying less loss, shortest delay or jitter, will be decreasing. Finally, such paths will be absent, so that additional users will not be allowed to network, as they anyway will not succeed to get enough quality for their transmission. In DDoS the analogous to AC approach is implemented, but from the destination side. Destination server allocates a portion of available resources to a client, before the latter starts to send data. If a client keeps transmitting rates below the claimed threshold, it will be safe. In the opposite case, it will be punished, so that destination server will inform all nodes in the path to block overhead traffic.

Random Neuron Networks is a universal tool for a vast range of applications. RNN's main advantage that it could be always described by Chapman-Kolmogorov balance equations, and if stationary distribution exists, it could be always found as product of neurons excitation probabilities. Another distinguishing property is learning. RNN can perceive the relationships between input and output variables by means of the Gradient Descent Learning and fix them into weights matrices. After learning process completion RNN will recognise the learnt patterns and produce the desired output. The Reinforcement Learning is implemented in network applications of RNN and allows to update routing decisions in the network nodes according to pre-defined QoS. Investigation in the field of extended RNNs is ongoing with purpose to work out such RNNs that will satisfy more complicated models.

1.3 Reinforcement Learning (RL)

It was described in the previous section, how CPN protocol uses RNNs in nodes and the product-from solution to make routing decisions, but the CPN also implements another property of RNNs, which helps to approach the desired QoS and is defined as Reinforcement Learning (RL). The RL algorithm changes RNN neuron potential by updating $w^+(i,j)$ and $w^-(i,j)$ weights, by means of rewards. Rewards are calculated from the values stored in the MailBoxes, which are updated by SPs(SACKs) and/or DPs(DACKs), according to pre-defined QoS. Thus, the connection between Reward and QoS "Goal function" G could be observed. The above mentioned Reward R is inversely proportional to the G, and general formula could be defined as follows:

$$R = \frac{\beta}{G + \epsilon} \tag{1.11}$$

where β is chosen from range $(0 < \beta \leq 1)$ and ϵ represents the lower bound for the QoS Goal value (could be a zero or some small value). When an ACK arrives to the node, it updates MB and triggers the reward recalculation, producing successive R_l value, $l = 1, 2, ...$ The next step of RL algorithm is threshold update that performed by exponential average, as follows:

$$T_l = \alpha T_{l-1} + (1 - \alpha) R_l \tag{1.12}$$

where α is a smoothing constant and is chosen on condition that $0 < \alpha < 1$. The value of previous threshold T_{l-1} is compared with current value of reward R_l for the RNN weights update. In fact, R_l is previous decision (previously selected neuron) reward, and T_{l-1} is a "historical value" of reward. If $R_l > T_{l-1}$, this case will significantly increase excitatory weights on links that lead to the previously selected neuron. At the same time inhibitory weights that lead to all other neurons in RNN will endure small increase. In case $R_l < T_{l-1}$, the previous decision neuron will be punished for not approaching the goal by significant increase of inhibitory weights going to it. Logically, all remaining neurons will be rewarded through some increase of excitatory weights leading to them. Thus, the punished neuron is "excluded" from the next decision making, but other neurons have

more chances to be chosen. The weights update is performed as follows:

$$
\begin{aligned}
&IF\ T_{l-1} < R_l \\
&\qquad w^+(i,j) \leftarrow w^+(i,j) + R_l \\
&\qquad w^-(i,j) \leftarrow w^-(i,j) + \frac{R_l}{n-2}, \qquad\qquad j \neq l \\
&ELSE \\
&\qquad w^+(i,j) \leftarrow w^+(i,j) + \frac{R_l}{n-2} \\
&\qquad w^-(i,j) \leftarrow w^-(i,j) + R_l, \qquad\qquad j \neq l
\end{aligned} \tag{1.13}
$$

To avoid weights overflowing the values in 1.13 should be re-normalised according to the forlmulae below:

$$
\begin{aligned}
r^*(i) &= \sum_{1}^{n} [w^+(i,j) + w^-(i,j)], \qquad\qquad i = 1...n \\
w^+(i,j) &\leftarrow w^+(i,j)\frac{r(i)}{r^*(i)} \\
w^-(i,j) &\leftarrow w^-(i,j)\frac{r(i)}{r^*(i)}
\end{aligned} \tag{1.14}
$$

In conclusion, this chapter presented the project objective and its main idea. The further chapters will give the detail understanding how idea was implemented in the CPN protocol code. The existing CPN background and theory was also provided in the chapter with purpose to assist the understanding of further chapters. All the abbreviations, which will be used in the project report, were provided in the chapter.

Chapter 2

Bilateral Traffic Differentiation

Existing CPN protocol and code realization, described in the previous chapter, is capable to search for paths that suit required QoS in the best way, track changes in the network conditions and update paths correspondingly. Nevertheless, there is a limitation that QoS is fixed and stamped by CPN sender node to any packet originated by it and could not be changed during transmission. In this chapter the CPN code enhancement is described that allows to change QoS during transmission depending on traffic volume in each direction of bilateral transmission. Moreover, one CPN sender node could allocate different QoS to different users that reside in it, so that packets with two distinct QoS are generated simultaneously from the same node. The aim of the enhancement is to make CPN network utilization more efficient, to load all available paths.

2.1 System Model

The bilateral network system model based on CPN is shown in Figure 1.1. It possesses two distinct QoS Goals for traffic differentiation, rather than the potential four distinct Goals for the two directional representations. Although the generalisation to four is straightforward, in this paper, we choose to limit ourselves to two distinct QoS Goals in order to simplify the experimental evaluation of the approach that we are proposing. The level of traffic volume in each branch is taken as a triggering condition between the two QoS Goals in each of the directions for each of the A and B pairs.

It is common in real life that the TX and RX traffic are asymmetric. An Uplink Sender could transmit to Downlink Sender much larger amounts of traffic than it receives,

and vice versa. Each sender should be able to estimate the ratio between its TX transmit and RX receive traffic levels and make a decision about the QoS for data that it sends. Obviously, if the sender is able to switch between QoS Identifiers, both QoS paths should be available to be used by sender. For this purpose the Smart Packets from each sender would discover each of the two QoS path simultaneously. As a result, each sender would possess at least two distinct QoS paths at any time and choose between them depending on the traffic conditions. Regarding the QoS Goals that we have selected, the smaller traffic level will be assigned the delay as the QoS Goal (QOS_DELAY), and the larger traffic would be allocated loss as the QoS Goal with path tracking (QOS_LOSS). For QOS_DELAY, CPN stays unchanged as in previous work: SPs discover paths; update them by reading timestamp information in Mailboxes, so that the shortest delay path is always available to the sender. QOS_LOSS is more complicated and consists of two distinct loss estimates: one is for Smart Packets, another for Dumb Packets. SPs' loss is used only in the intermediate nodes of CPN - i.e. those that are used in the CPN cloud in Figure 1.1 in order to discover and evaluate paths. Each intermediate node estimates the loss of Smart Packets that leave this node heading towards destinations; the RNN in the node evaluates the success of the node for each destination that the Smart Packets search. As in CPN, the DPs' Loss evaluation can only be determined at the sender node (A for a flow from A to B) since it requires that the number of DACKs returning to the sender be counted and compared to the number of DPs that have been sent out to that particular destination. Naturally, when QOS_LOSS is used, the loss measurements are being carried out on the path that is offering the smallest loss rate since that is the path that is being used. Besides, the QoS decision and the path choice, each sender actually must manage flow control. When a sender has several destinations, it must distinguish between them and forward packets while being aware of the distinct flows, as shown in Figure 2.1.

2.2 QoS Decision Logic and Flow Control

The QoS decision logic and the Flow Control scheme, both of which reside in sender nodes, are described in detail in this section. The Flow Control algorithm provides TX and RX traffic rate estimation per destination for the QoS decision logic.

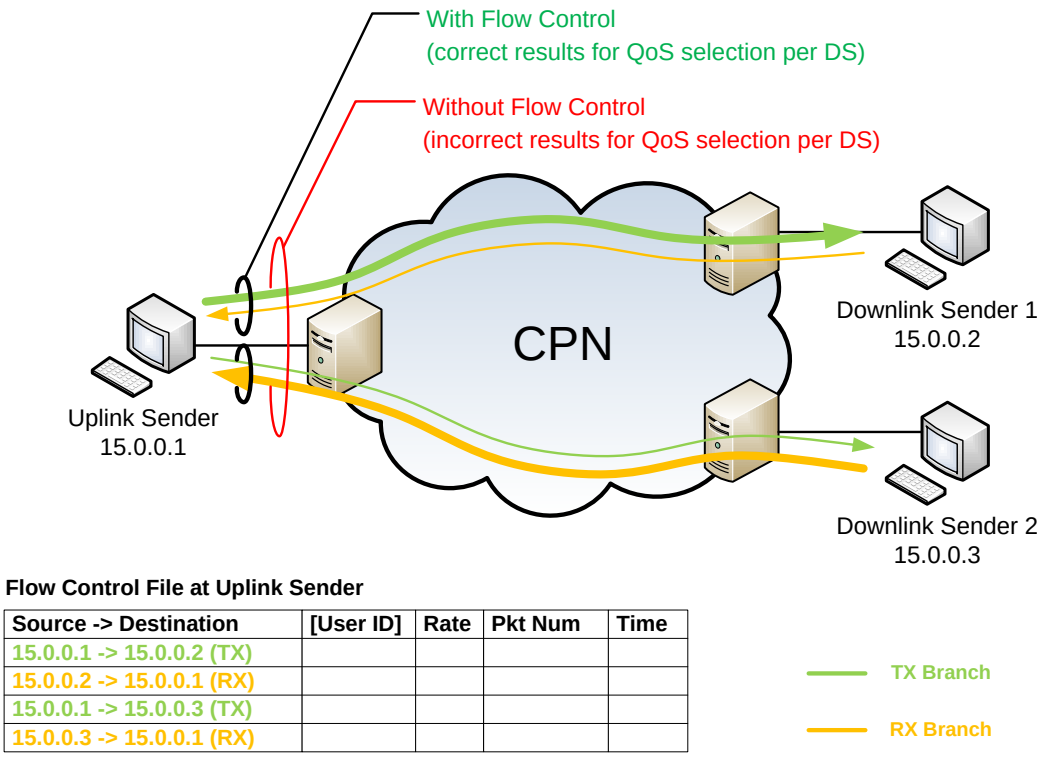

Figure 2.1: The Flow Control Logic for QoS decision making.

2.2.1 Round-robin for Smart Packets

Before further description about decision making the path discovering technique should be mentioned. Uplink Server always initialises the path searching algorithm towards Downlink Sender. DS starts to search for the path towards US only after receiving packets from US. As there are two QoS in the system, each sender needs to discover paths for every QoS. When each sender tries to send a SP, it needs to mark it with QoS Identifier. For simultaneous discovering of paths for each QoS, the round-robin principle is applied for QoS Identifier selecting. If the previous SP was marked with QoS1, the current one will be marked with QoS2. After that the cycle will be repeated. Thus, 50% of all SPs will be travelling with QoS1 and another 50% - with QoS2.

2.2.2 Flow Control

After obtaining to distinct QoS paths each sender can start DPs transmission. Each sender sends DPs towards destination; these packets are regarded as Originated Dumb Packets. At the same time each sender terminates DPs from destination; these packets are called as Locally Delivered Dumb packets. Only these packets trigger the Flow Control functions. Thus, the intermediate nodes are not loaded with this logic, as they just pass DPs through. Flow Control file stores information about both branches (TX and RX) per destination and has following fields (see Figure 2.1):

- **Source → Destination**. This field consists of two IP addresses of each branch that a sender has. TX and RX branches have opposite directions of traffic flow. Thus, local server address will be put into Source sub-field for TX branch and into Destination sub-field for all RX branch. Destination Server IP address resides in the Destination and Source sub-fields respectively. There are two records per destination.

- **User ID**. The value in the field indicates which user generates RX or TX traffic. If there are two users sending from the same source two the same destination, there will be two records TX traffic and two records RX traffic in the bilateral transmission case. To allow the Flow Control algorithm to distinguish packets by originating user, a field was added to the DP header that was defined as **User ID** (see Figure 1.3). The Flow Control logic reads the **User ID** field and textbfSource, Destination fields from DPs and DACKs headers and decides which **Packets Number** fields to increment in the Flow Control File.

- **Rate**. The packet rate of the branch is written into this field.

- **Packets Number**. This field accumulates DPs. Originated DP increments the Packet Number field of respective TX branch. Locally Delivered DP increments the Packet Number field of respective RX branch.

- **Time**. This field tracks the time as the rate is calculated every 2 seconds.

It is obvious from fields' description that File Control File counts the number of DPs sent and received in corresponding TX and RX branch per each destination server. Every

2 seconds the packet rate is calculated and written into the designed field. The Rate field values are used in the QoS Decision Logic. The Flow Control file update is not bounded to any QoS. It is launched by any DP not considering the QoS mark in DP's header. Nevertheless, the Flow Control Logic differentiates traffic generated from the node in terms of users who send packets. Each user traffic is tracked separately from others. Moreover, QoS Decision Logic makes decision about each user QoS independently. As it was mentioned before users are defined by their unique identifiers. The User IDs allocation solution in this project was straightforward, IDs were defined by Linux OS users numbering. Thus, one CPN sender node can generate two distinct traffic flows with two distinct QoS, if there are two Linux users sending from distinct User Spaces of OS.

2.2.3 QoS Decision Logic

SPs QoS marking was already described and reminds Round-Robin scheduling. DPs QoS selecting and marking is formed into QoS Decision Logic. It directly depends on the level of asymmetry of traffic volume between TX and RX branches. Before Originated DPs sending each server should decide which QoS to select for adding into DP header. The simple function performs comparing of TX and RX branches rates of destination towards which packets should be sent. The mentioned rates are retrieved from Flow Control File. Then the proportion of TX traffic in the Sum Traffic is computed according to formula:

$$Ratio = \frac{TXRate}{TXRate + RXRate} \tag{2.1}$$

If Ratio is more than 0.6 (it means that TX traffic comprises 60% of overall traffic travelling between source and destination), the less loss QoS with path tracking will be assigned to the DPs. The marking DP with QoS means that it will follow all rules defined by the given QoS: path selection, interacting with MBs in intermediate nodes, files updates and etc.

2.3 Less Loss QoS with path tracking

Less Loss QoS with path tracking differs from previous QoS formats, because it consists of two functionally independent loss estimation algorithms. The first algorithm estimates

SP Loss per destination and runs in intermediate nodes. The second one estimates DP Loss per path and runs in sender nodes.

2.3.1 Smart Packet Loss QoS in Intermediate Nodes

When SP searches for the path, it can do it randomly or using Reinforcement Learning (RL). SPs with random search are 5% of all packets sent by sender and SPs with RL are 20%. Random search logic is straightforward. Each intermediate node after receiving a SP selects the next hop for it among his available neighbours randomly. In RL intermediate node uses Random Neural Network (RNN) [12]. For Delay QoS RNN is created in each node following the previous versions of CPN. There is a distinct RNN per combination of Source Address, Destination Address and QoS Identifier. For SP Loss QoS in intermediate nodes RNN is created per combination of Destination Address and QoS Identifier and shows the success of the node in path searching. Nevertheless, the structure and logic of RNN should remain the same. To achieve it, the Source Address was assigned with Zero IP Address (0.0.0.0). Thus, the anchoring of RNN to the source address was eliminated.

Analogous changes were introduced in MB format from which RNN takes values for neuron weights update. Each entry in the MB is created per destination address and QoS ID (head of the MB), and there are all neurons listed under each entry. Each neuron entry has the following fields (see Figure 2.2):

- **Neuron Address**: This field contains the IP Address of the neuron.
- **SPKTs**: This field accumulates the number of SPs that travel through this neuron towards destination from the MB head.
- **SACKs**: This field accumulates the number of SACKs that travel through this neuron from destination from the MB head.

To prevent the overflow in counter fields, after every 15 sec SPKTs and SACKs fields are set to zero. This also grants the memoryless property to the system - the estimation does not count the previous loss towards destination that no longer exists.

The fields that count SPs and SACKs are incremented when a packet of respective type travels through the node. The neuron address to which SP is routed or from

Zero Addr → Dest Addr	[QoS ID]	
Neuron (0) Addr	SPKTs	SACKs
Neuron (1) Addr	SPKTs	SACKs
...	...	...
Neuron (N-1) Addr	SPKTs	SACKs

Figure 2.2: Mailbox Format for SP Loss QoS in intermediate nodes

which SACK has arrived defines exact SPKTs or SACKs field that should be incremented. Rewards for RNN are calculated from these fields according to the following formulae:

$$Loss_{SP} = \frac{SPKTs - SACKs}{SPKTs} \tag{2.2}$$

$$Reward = \frac{\beta}{Loss_{SP} + \epsilon} \tag{2.3}$$

The parameters in (2.3) have the following values: $\beta = 0.5, \epsilon = 0.0001$.

2.3.2 Dumb Packet Loss 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 storage file was designed. Each entry in this file stores one path and has the following fields:

- **Destination Address**: This field contains the IP Address of destination towards which a path was found.
- **Active**: This field shows whether the path is active now. Path is active when DPs are sent over it. If path is marked as "active" in this file, it means that the same path is stored now in DPRR. The field has two possible values"0" - for inactive paths and "1" - for active path.
- **DPKTs**: This field accumulates the number of DPs sent over the given path towards

destination from previous field.

- **DACKs**: This field accumulates the number of DACKs sent over the given path towards destination.

- **Path ID**: This field contains the unique identifier of the path. It helps easily and fast to identify the path through which a DP was sent or a DACK was received. When DP is created by a sender it is marked with the Path ID. The corresponding DPKTs field of the selected path is incremented. For purpose of marking the additional field was introduced into the packet header and called **Route ID**. Upon DACK packet receiving sender reads this field and defines the DACKs field of which path to increment. To prevent the overflow in counter fields after every 180 000 DPs sent over a path, DPKTs and DACKs fields are set to zero. This also grants the memoryless property to the system - the estimation does not count the previous loss in the path that no longer exists.

- **Path**: There is the whole path stored in the field. It is used when a new discovered path is checked for uniqueness.

The algorithm with path tracking works as follows:

1. When DPRR and Path Control File are empty, any discovered path is written directly into DPRR and Path Control File and becomes "active" in the latter one. DPs starts to use path immediately and starts to increment the counters.

2. When a new path is brought by SACK it should be checked for uniqueness. If there is no completely the same path in the Path Control File it will be immediately written into DPRR and Path Control File and becomes "active" in the latter one. It was designed this way with purpose to test new discovered paths for DP Loss. If brought path is not genuine, it triggers the algorithm of selecting the path with less DP Loss.

3. If path is not genuine, there are already some values in DPKTs and DACKs fields for this path in the Path Control File. The DP Loss of this path that was brought by the latest SACK could be calculated. At the same time the DP Loss of the current "active" path should be 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. The DP Loss is calculated as follows:

$$Loss_{DP} = \frac{DPKTs - DACKs}{DPKTs} \tag{2.4}$$

4. It could be a case when two compared paths are the same. It means that algorithm has reached a convergence for some period of time, so that the less SP Loss path and less DP Loss path are equal.

In conclusion, this chapter provides information about programming updates and enhancements introduced to the existing CPN code with purpose to increase the efficiency of CPN network utilization. The introduced features are Round-Robin for SPs for two distinct QoS simultaneous path discovery, users traffic differentiation, the Flow Control Logic for user's TX and RX traffic volume asymmetry tracking, the QoS Decision Logic, Less Loss QoS with path tracking. In the next chapter, the updated CPN code was checked for performance by means of experiments.

Chapter 3

Experiments and Results

This section presents the experiments description that were performed on the enhanced CPN network and results of these experiments with analysis.

3.1 The CPN testbed and Tests Scenarios

The CPN code was implemented and run on Pentium server machines with Linux Ubuntu operating system installed on them. The kernel version was 2.6. The CPN testbed network configuration is given in Figure 3.1. All links between nodes are 100 Mbps Ethernet links (12.5 MBps). Experiments were divided into two groups. The first group of experiments was designed for one user case per node, with two CPN nodes involved in bilateral transmission link. The aims of the first group tests were:

1. To confirm that the program code is working as designed in the scope of capability to allocate QOS_LOSS to larger traffic direction, to discover and track all available paths.

2. To evaluate the DP Loss vs Traffic Rate.

3. To evaluate the percentage of paths usage for Uplink Branch by QOS_LOSS traffic.

The main idea of the second group of experiments was to simulate four QoS case. There were two distinct users per node with two CPN nodes involved into bilateral transmission link. Four users were organized in pairs, so that each user in each sender node communicates with only one user in another sender. Each node possessed both QoS algorithms of path discovery and users could select the QoS depending on the traffic conditions. The test

was designed with purpose that two users in one sender node are allocated with different QoS. The aims of the test were:

1. To confirm that the program code is working as designed in the scope of capability to treat each user's traffic independently from other users in the same sender node.

2. To evaluate ability of the algorithm to exploit users diversity for efficient CPN network resources utilization.

3. To measure the percentage of time when two distinct QoS algorithms select simultaneously the same path for DPs transmission from each sender perspective.

4. To measure and evaluate the number of distinct simultaneous paths in the network.

5. To evaluate and compare the percentage of paths usage by each sender node.

6. To evaluate and compare the percentage of paths usage by each QoS.

7. To evaluate the percentage of paths usage by whole CPN network.

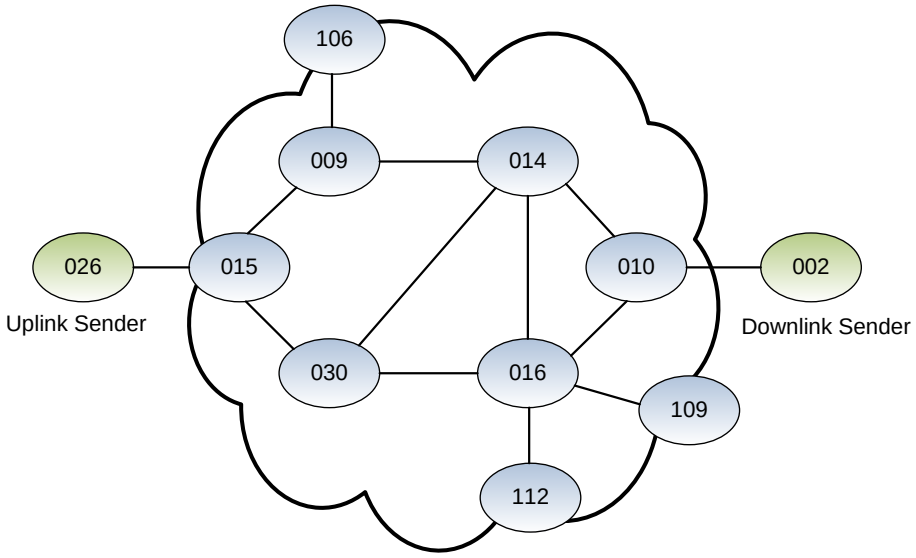

Figure 3.1: The CPN testbed used in the experiments

All experiments were run for variety of traffic rates. As tests for every rate were run separately from others, Path IDs, tracked by Path Control File, could not match. It

could not be predicted which path would be discovered first and would get the smallest index. To be able to compare paths for different rates and experiments, the unification was needed so that the Path Legend was designed (Table 3.1). The path enumeration from the Legend was used and for plots and graphs creation to make process of comparison and analysis possible.

Path ID	Route								Length
0	026	015	009	014	010	002			Short
1	026	015	030	014	010	002			Short
2	026	015	030	016	010	002			Short
3	026	015	009	014	016	010	002		Medium
4	026	015	030	014	016	010	002		Medium
5	026	015	030	016	014	010	002		Medium
6	026	015	009	014	030	016	010	002	Long

Table 3.1: Paths Legend

The Matlab programming environment was used to interpret experimental results. The programmed scripts performed files and logs parsing and automatic calculation of loss and paths usage. Plots and bar diagrams were also produced by Matlab scripts. Using of Matlab programmed scripts allowed to collect enough experimental data.

3.2 Experiments with One User per Sender

During this block of experiments traffic was generated from Uplink Sender 026 towards Downlink Sender 002. Traffic from US was much larger then from DS and varied. There were two tests scenario. In the first scenario traffic was generated in the range 100 - 1000 pps (100, 200, 300, 400, 500, 600, 700, 800, 900, 1000) with packet size 1024 bytes (1 Kbyte). Due to the rates involved this scenario was regarded as Low Traffic scenario. In the second scenario traffic was generated in the range 1000 - 12500 pps (1000, 2500, 5000, 7500, 10000, 12500) with packet size 1024 bytes. Naturally, this scenario was named Large Traffic Scenario. Preliminary test running showed that real packet rate upper-bound was 10750 pps that corresponds to the 10.75 MBps rate, and it did not increase even though 12500 pps rate was generated. Thus, the CPN protocol overhead is $(1 - 10750/12500) = 0.14 = 14\%$. Nevertheless, the DP Loss for pre-defined 12500 pps rate was observed to be larger than for

pre-defined 10750 pps. According to this DP Loss plots and Paths Usage plots maximum rate value was left equal 12500 pps, even though sender node was, in fact, sending packet at the 10750 pps rate.

The experiments show that CPN code managed to discover all seven available paths during each test run and to perform path loss tracking. The results of the experiments are represented in plots of DP Loss vs Rate and Percentage of each path usage vs Rate. Results were collected into the log that was recording the selected Path ID and Path Loss, after a decision about path was made by Less DP Loss Logic with Path Tracking. Naturally, for higher rate there was a larger number of decisions during the time of test duration. The resulting figures are averaged over all number of decisions, occurred during the test run.

3.2.1 Low traffic One User Scenario Results

Duration of one Low Traffic scenario run was 3 minutes. The scenario was run 5 times for every rate. Results are given in Figure 3.2 and 3.3. In the Figure 3.2 the plot of Loss Percentage for rates 700, 800, 900 and 1000 is given. For rates less than 700 pps the nodes were not saturated and DP loss was always zero. The saturation is assumed to begin slowly from rate 700 pps (= 0.7 MBps). The Less DP Loss Logic with path tracking sometimes introduces some error while estimating DP Loss for path. It could count DACK as lost, when it is just delayed. Nevertheless, if DACKs start to endure delays, it means that path saturation is increasing, and it is smarter and better to switch to the path with less or even zero loss. The DP Loss approximation has some error, but it does not affect the right way of the algorithm functioning. Moreover, the DP Loss results approximated by path tracking code were compared with the statistical data of exact number of lost DACKs. The approximation is very close to the real loss numbers.

The percentage of path usage given in Figure 3.3. The most used paths at all rates after the results analysing are Path_0 and Path_2. Path_0 is the most used with 40% on average for rates from 700 to 1000 pps, Path_2 is the second with 25% of average usage. The high usage of these paths could be easily explained that they are the shortest paths. Nevertheless, there are three shortest paths of equal length (Path_0, Path_1, Path_2), and the Path_1 usage is very low, about 3% on average. The Path_4 usage does not

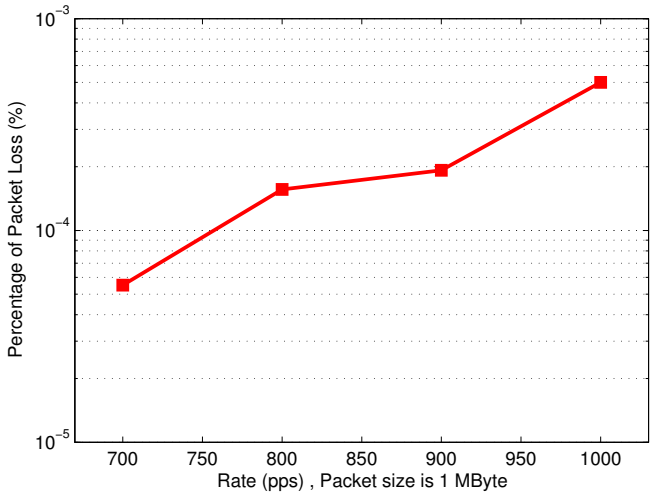

Figure 3.2: The Loss percentage plot for Low Traffic Test scenario.

supersede Path_1 significantly, approximately 4%, even though it is a medium length path. It is remarkably that there is the longest length Path_6 among three paths, which share third place in usage percentage with 10% on average. The presence of medium length paths, Path_3 and Path_5, is logical, but Path_6 shows an outstanding performance in approaching QOS_LOSS goals.

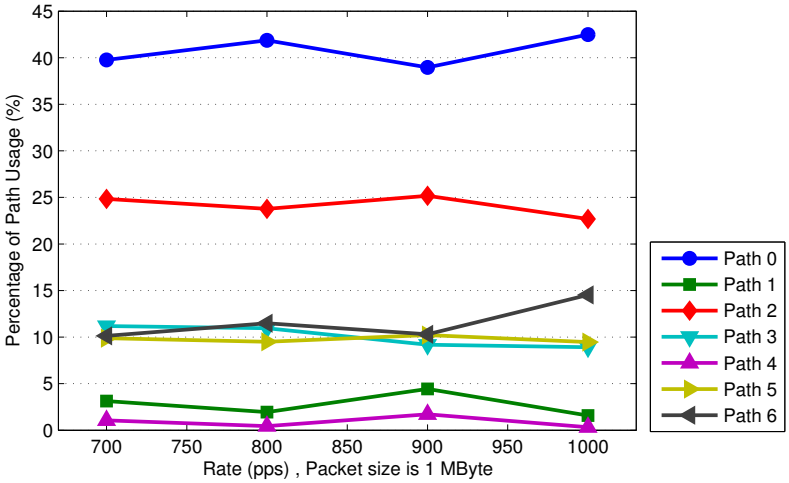

Figure 3.3: The percentage of path usage for Low Traffic Test scenario

To conclude, the system is obviously is not overloaded according to the test results. There is a stable and low percentage of DP loss. From the path usage results it could be summarised that the best performance was showed by Path_0. The longest length

Path_6 occasionally was as good as paths with medium length, like Path_3 and Path_5. The less used paths are Path_1 and Path_4. The Low Traffic test showed that the code is working and fulfils its tasks. Nevertheless, the network load was too low and could be not representative for QOS_LOSS performance and paths usage distribution. The Large Traffic scenario was designed and run with purpose to clarify trends.

3.2.2 Large traffic One User Scenario Results

The Large Traffic experiments were organized as in the previous scenario. One test run duration was 3 minutes, and there were 5 runs for every rate. Results are given in Figure 3.4 and 3.5. The upward trend of Loss percentage in Figure 3.4 shows that during the test system was saturated. The level of saturation increased with packet rate. The trend rockets after 2000 pps rate, but DP loss is about 0.01% until rate 8000 pps. For 12500 (12.5 Mbps) rate DP Loss is equal 0.25%. The DP Loss percentage could be regarded to be low, but it should not be forgotten that the network is small with large interfaces bandwidth. The another option is that it is the effect of DP Less Loss Logic functioning. In fact, the most important thing is the trend behaviour. The DP Loss increase is significant for high packet rates, comparing to the lower rates.

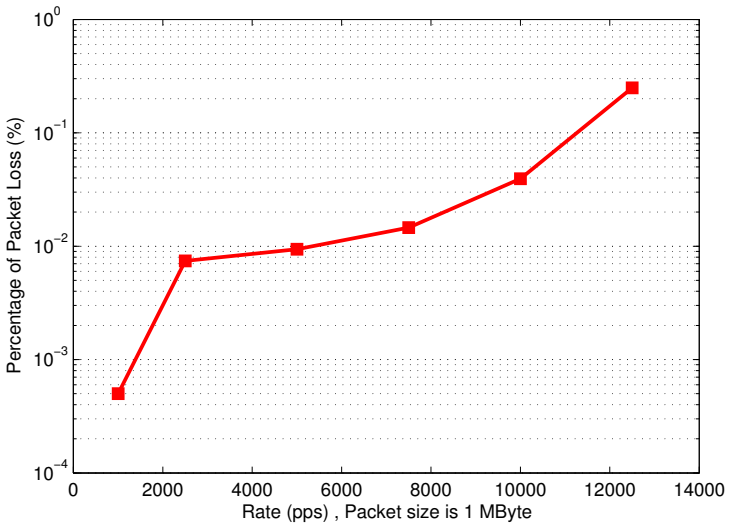

Figure 3.4: The Loss percentage plot for Large Traffic Test scenario

The path usage plot is given in Figure 3.5. The results highlight the inconsistency of estimating best path only by the shortest length for distinct QoS Goals. The best

performance paths are Path_0 with 40% usage and Path_2 with 25% that are the shortest paths. Path_6 is at the third place with approximately 13% usage on average. The Path_6 in this experiment excel medium length paths, Path_3 and Path_5 with 10% of usage. The usage percentage of Path_1 and Path_4 is almost equally low with 2-3%. Nevertheless, for rates over 8000 pps the trends stops being flat. The less used paths trends go upwards, and the usage percentage of popular paths starts to decrease. It could be explained that at high packet rates saturation introduces a large congestion, so that algorithm seeks for the unused resources to maintain the performance.

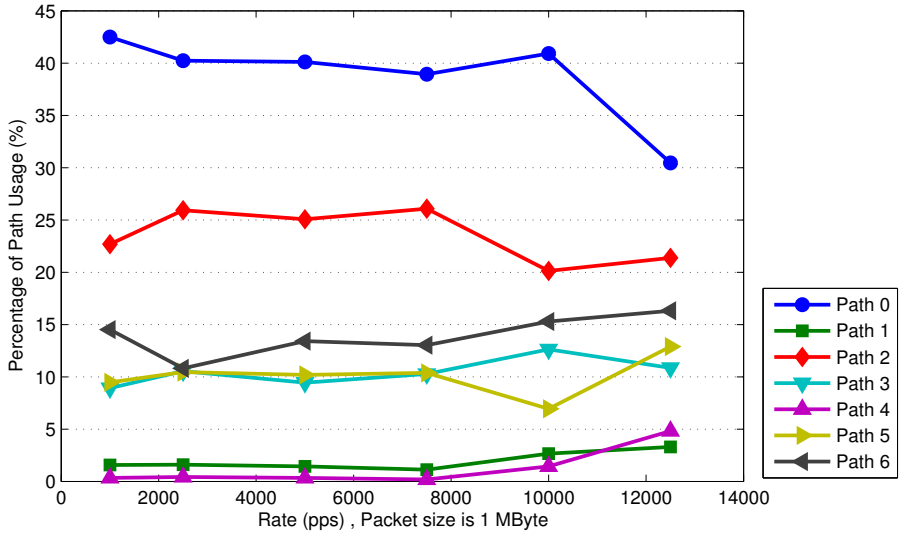

Figure 3.5: The Loss percentage plot for Large Traffic Test scenario

To conclude, the Large Traffic test scenario confirmed the rankings of DP Less Loss paths with Path_0 at the first place, Path_2 - at the second, and Path_6 has eventually managed to supersede shorter length paths and to get the third place. Path_1 and Path_4 continued to be the less used paths, but at the high rates the usage these paths started to increase. In the high load condition the algorithm tries to use all available resources effectively. It took part of the load from the most used paths and allocated it to the less used paths to decrease the congestion and DPs Loss. The values of DP Loss shows that the algorithm manages to maintain very low loss packets by smart paths selection.

3.3 Experiments with Two Users per Sender

This group of experiments uses two users pairs from the previous experiment to simulate the four QoS transmission network. Two distinct users resided in the cpn026 node and two distinct users - in the con002 node. There is a scheme of the experiment, given in the Figure 3.6. The Linux OS User spaces were utilised to generate two distinct flows from each CPN sender node. The first user was *root* with User ID 0 and the second user was *cpn* with User ID 1000. The users were organised in pairs. The first pair: user *cpn* (1000) in cpn026 was sending large rate Uplink traffic to user *root* (0) in the cpn002, the latter responded back with low rate Downlink traffic. The second pair: user *cpn* (1000) in cpn002 was sending large rate Uplink traffic to user *root* (0) in the cpn026, the latter responded back with low rate Downlink traffic. Referring to the section of QoS Decision Logic, the CPN code in each node would allocate different QoS to its Uplink and Downlink branches that it maintain. In other words, each user in every node would be assigned with different QoS from another user in the same node.

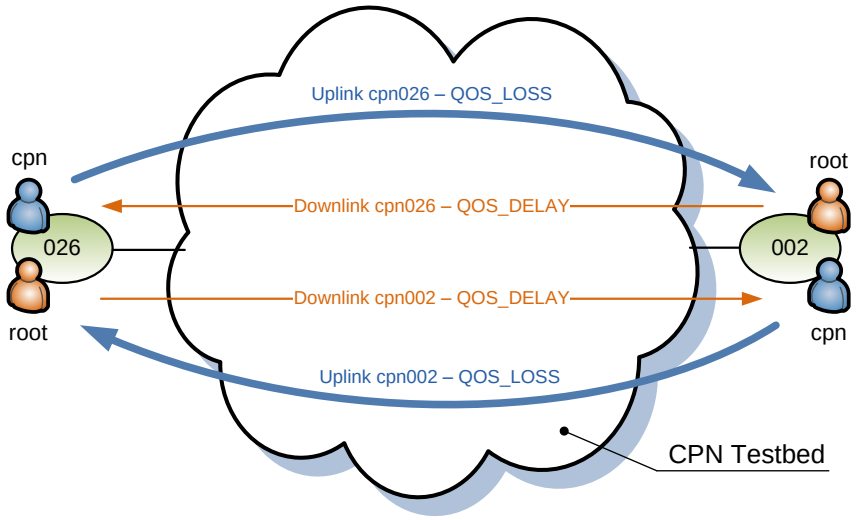

Figure 3.6: The test scheme for 4 QoS Simulation (two users per sender node)

To be able to see the effect of QOS_LOSS functioning, the rates range was chosen to be from 1000 to 10000 pps (1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000). The rate values were assigned to be equal at each sender node. If *cpn* generates

1000 pps traffic from cpn026 node, then *cpn* user in the cpn002 also generates 1000 pps. Thus, traffic rate that travel through CPN network was twice the traffic rate, generated by one *cpn* user. The *root* user's traffic was not larger than 100 pps. The test outline was organized into test runs of 3 minutes duration with 5 independent runs for each rate. Upon tests completion results were averaged for each rate. The results were collected as reading DPRR file, which stores two active paths for every QoS, and copying it contest to the output text file every second. Then text files were parsed by Matlab scripts and the plots were created.

3.3.1 The same path usage results

In this subsection the percentage of time of the simultaneous same path usage by two distinct QoS, as it is seen from each sender node, are provided. The plots are given in the Figure 3.7 and Figure 3.8. The percentage of time, when the same path is simultaneously selected by QOS_DELAY and QOS_LOSS at the cpn002 sender, varies in the range 16 - 25%, so that $\approx$ 80% of time DPs from cpn002 travel different paths. The percentage of time, when the same path is simultaneously selected by QOS_DELAY and QOS_LOSS at the cpn026 sender, varies in the range 15 - 21%, so that again $\approx$ 80% of time DPs from cpn002 travel different paths. For all rates range CPN code algorithms manage to maintain different path usage percentage almost at the same 80% level, as the trends fluctuations for both senders are small, $\pm$ 4.5% around average of 20.5% for cpn002 and $\pm$ 3% around average of 18% for cpn026.

To sum up, the experiment results show the effectiveness of enhanced CPN code in the bilateral traffic differentiation. Each CPN sender manages Uplink and Downlink independently be means of two distinct QoS, allocated to each branch. In the 80% of time *cpn* user DPs and *root* user DPs from each sender travel different paths.

3.3.2 The evaluation of distinct paths number in the CPN network

The previous subsection results analysis showed that in $\approx$ 80% of time DPs with different QoS, generated by one sender, travel different paths. The maximum number of simultaneous paths in the CPN network is upper-bounded by number of traffic flows and equal to 4. This subsection test results present the percentage of time, when there are N distinct

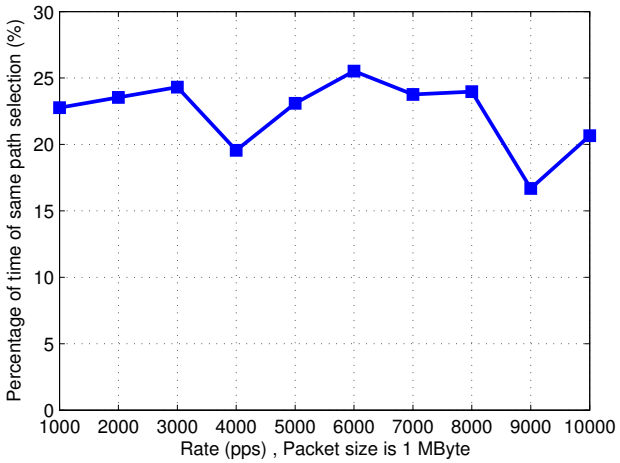

Figure 3.7: The percentage of time two distinct QoS algorithms select the same path for DP transmission from cpn002

simultaneous paths in the network. Variable N is defined in the range from 1 to 4. When N is 1, it means, that all four users have selected the same path for DPs sending. Even though the DPs travel in the opposite directions, they can utilise the same path. When N is 2, there are two possible cases. Two users have selected the same path (one from cpn002, another from cpn026 or both users from one of the sender) and remaining two users selected the same path, but different from the previous two users' path. Another case is when three users have selected the same path and one user - different path. When N is 3, it means that three users has selected three different paths, but path, selected by forth user's, coincide with one of three users. Finally, the ideal case is when all four users have chosen distinct paths. The bar diagram of percentage of time, when there are N distinct simultaneous paths in the Network, is given in Figure 3.9. The diagram shows that the enhanced CPN protocol has a stable performance for all rates. The most frequent number of paths in the CPN network for all rates is 3, in more than 50% of time. The percentage of time for 2 and 4 paths varies with rates, but it is remarkable that in $\approx 70\%$ of time on average there are more than 3 paths in the network (3 or 4). The presence of only one path is enormously rare, and does not occur more than in 1-2% of time.

The probability distribution function of simultaneous distinct paths in the network is given in the Figure 3.10. The histogram represents the distribution, obtained from

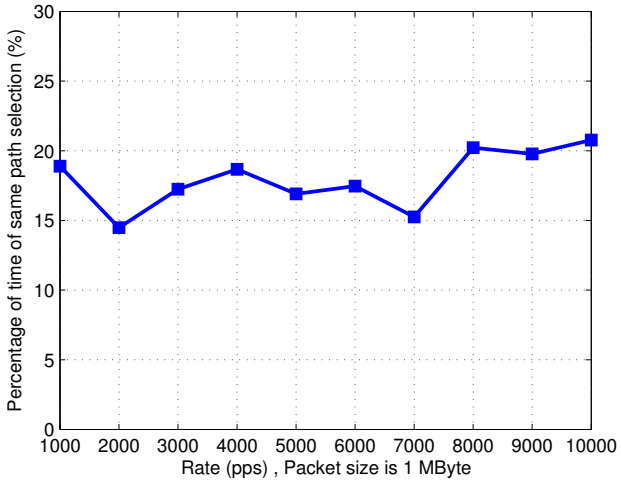

Figure 3.8: The percentage of time two distinct QoS algorithms select the same path for DP transmission from cpn026

empirical data, and plot shows the theoretical approximation. The empirical distribution is very close to the normal distribution with mathematical expectation equal 3 and standard deviation equal 0.75. There are 3 paths in the network in 52% of time, 2 or 4 paths - in 22% of time, and only 1% of time there is one path.

In conclusion to the presented results, the enhanced CPN code manages to archive target of efficient CPN network resources utilization through bilateral traffic differentiation. The tests showed that at least 70% of time there will be 3 or more distinct simultaneous paths in the network. The percentage of the worst case, when there is only one path, chosen simultaneously by four users, is almost negligible and equal 1%.

3.3.3 The Paths Usage in the 4 QoS simulation

This subsection describes experiments results of paths usage percentage. The experiment outline allowed to represent paths usage in three scopes. First, the Figure 3.11 and 3.12 provide the plots of the most selected paths as it seen from each CPN sender node. It means that the paths usage was evaluated form each sender node, not considering the QoS that were selecting the paths.

The paths usage results for cpn002 are given in Figure 3.11.The cpn002 node prefers, as expected, Path_0 with usage percentage of 40% on average, until saturation begins at 7500 pps that decreases the Path_0 usage till 27-30%. The second place of usage

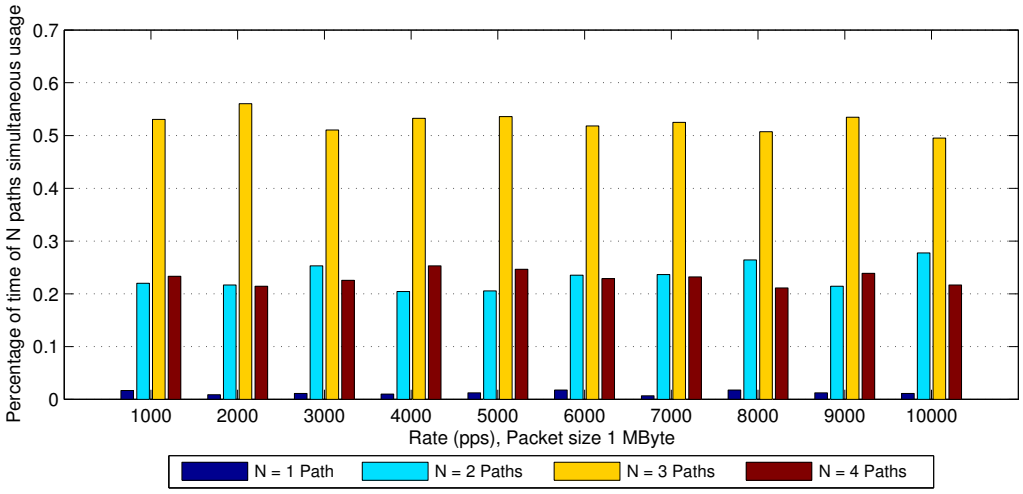

Figure 3.9: The percentage of time of N simultaneous paths in the CPN network vs Rate

is divided by two other shortest paths Path_1 and Path_2 with average about 17%. The Path_1 performance is stable, as the plot is flat, but the usage of Path_2 increases with saturation raising and becomes equal to Path_0 at the rate 10000 pps. The Path_6 is used more often then shorter length paths, it shares the third place of usage with Path_5 that is used on average in 7-8% of time. The less used paths are medium length paths Path_3 and Path_4 with $\approx 4\%$ of average usage, but their performance is improved with rate increase. After 9000 pps the less used paths are at the same level as Path_6 and Path_3.

The paths usage ranking from cpn026 has a different outline. The Path_0 is no more an absolute leader, it shares the first place with Path_2, both having 27-28% of average usage (see Figure 3.12). The four paths with variety of lengths are used on average in $\approx 10\%$ of time, they are short length Path_1, medium length Path_3 and Path_5 and the longest Path_6. The saturation introduces some ordering among these paths at rates after 8000 pps, as follows: Path_3, Path_1, Path_6 and Path_5, but it does not significantly change the values of usage. The Path_4 is less used path with 4% of average usage.

The comparison of paths usage percentage from each CPN sender node shows that the most popular path is Path_0. Nevertheless, Path_0 and Path_2 are used at the same level, but another short length Path_1 is used 3 times less. The cpn002 node makes Path_0 absolute leader, by using Path_1 and Path_2 at the same level, but ≈ 2 times less than Path_0. Other paths are used by both nodes almost equally. The remarkable performance

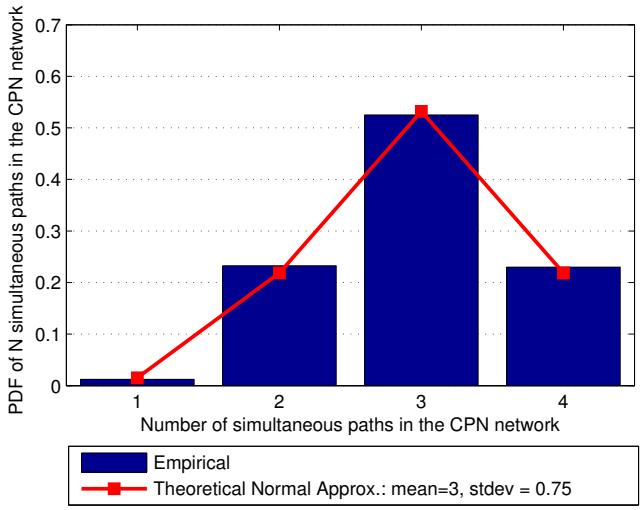

Figure 3.10: The PDF of N simultaneous paths in the CPN network

is shown by Path_6 again, as it never at the last place in usage. The last place are always occupied by Path_4. The saturation does not introduce a lot of changes into paths usage rankings of cpn026, but it does for cpn002. For the latter at the high rates Path_0 and Path_2 usage becomes equal.

For the second scope the Figure 3.13 and Figure 3.14 provide the plots of the most selected paths as it seen by each QoS. It means that the paths usage was evaluated for each QoS, not considering from which CPN sender packets were originated. The opposite direction paths were regarded as the same path, if packets in one direction travel through the same nodes as in the opposite direction.

The paths usage plots for QOS_DELAY are given in Figure 3.13. The Path_0 is the most used path with 30% of average usage, the second place occasionally goes to Path_1 with $\approx$ 23% average usage until saturation, the Path_2 is at the third place with 17%. The fourth place of usage is shared by all remaining paths, each path usage varies form rate to rate independently from others, but it does not go out of range 5-10%. The Path_4 is also among these paths, it is no more a less used path in QOS_DELAY results. The saturation influences the rankings by allocating DPs almost equally to Path_1 and Path_2, so that each of them is used in 16-17% of time at rate 10000 pps. At the same rate Path_3 improves its performance, and its usage increases from 7-8% twice, up to 14%.

The paths usage plots for QOS_LOSS are given in Figure 3.14. The Path_0 is

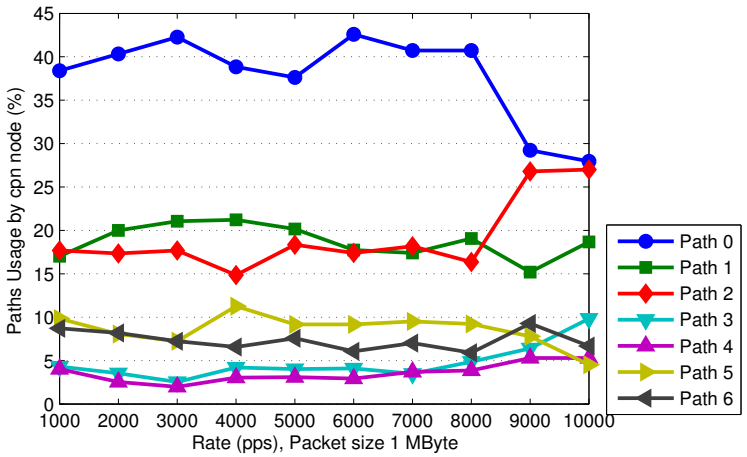

Figure 3.11: The percentage of Paths Usage by cpn002

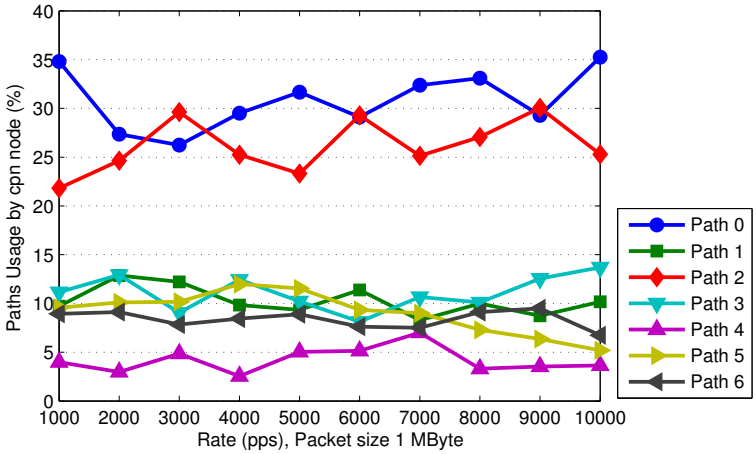

Figure 3.12: The percentage of Paths Usage by cpn026

preferred by QOS_LOSS and selected on average in the 40% of cases. The second most used path is Path_2 with 25% of average usage. The medium Path_5 and longest Path_6 share the third place, as they are used on average in 10%. The short Path_1 and medium Path_3 show a little bit worse performance with 5%. The Path_4 is almost completely unused by QOS_LOSS. The usage percentage at all rates almost zero, so that Path_4 could be named a worst path in approaching DP Less Loss Goal. The high rates over 8000 pps bring changes into paths usage. Usage of the Path_0 decreases, at the same time Path_2 usage increases, so that their usage become equal at average 35% per path. The Path_6 and Path_5 group interchanges in position with the Path_1 and Path_3 group. The

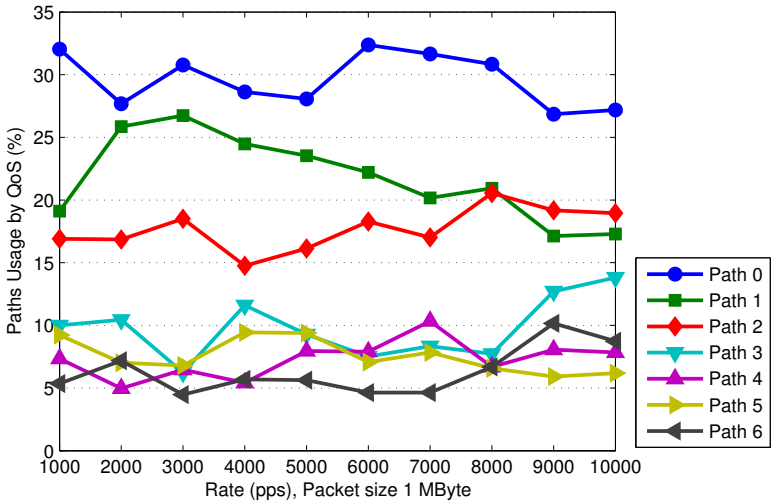

Figure 3.13: The percentage of Paths Usage by QoS DELAY

first group has 5% of usage per path at 10000 pps rate, the second group has 10% per path.

The QOS_DELAY and QOS_LOSS have showed different preferences in the paths selection and usage frequency. Two most used paths Path_0 and Path_2 are equal for both QoS, but the Path_1 is preferred by QOS_DELAY more, than by QOS_LOSS. Thus, QOS_DELAY uses three shortest paths to balance traffic, QOS_LOSS appreciates only two of them Path_0 and Path_2. The usage of remaining paths by each QoS is approximately at the same level, except Path_4. The QOS_LOSS tried to exclude Path_4 from its DP transmission, when QOS_DELAY uses it on average in 10% of time. The longest Path_6 is also used by each QoS, confirming the fact that the path length is not the main factor in approaching distinct QoS Goals.

Comparing two scopes, paths usage by CPN sender and paths usage by QoS, it could be noticed that cpn002 behaves according to QOS_DELAY and cpn026 tries to follow QOS_LOSS behaviour. Of course, this statement means that CPN nodes behaviour just reminds the behaviour of QoS, but analogy is indicated by Path_1 usage. The cpn002 node and QOS_DELAY used it quite often, at the same time the cpn026 and QOS_LOSS appreciate Path_0 and Path_2 and balance most of the traffic between those two. Nevertheless, the cpn002 gives 40% of average usage to Path_0, like QOS_LOSS, and the cpn026 selects Path_0 is on average in 30% of time, like QOS_LOSS.

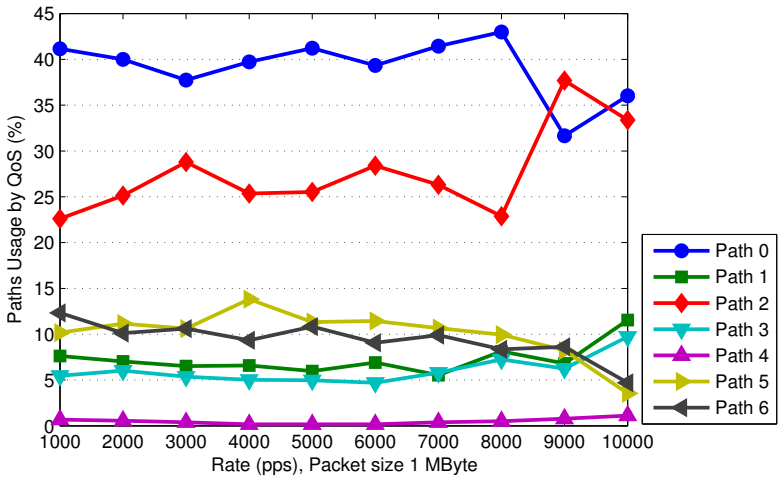

Figure 3.14: The percentage of Paths Usage by QoS LOSS

In the third scope, the Figure 3.15 provide the plots of the most selected paths in the whole CPN network. The combination of paths preferences from two sender nodes and two QoS are presented here. The Path_0 has average usage of 35% until 8000 pps rate and, as expected, the most used path in the CPN network. The average usage of 22% has Path_2, and it is at the second place. The third most used path is Path_1 with on average 15%. The forth place is shared by Path_6, Path_5 and Path_3, each of these paths has average usage in the range 8-10%. The less used path is Path_4 with average usage of 4%. The high rates make the Path_0 usage to decrease and Path_2 to increase, so that there is $\approx$ 28% of usage per path for rates over 8000 pps. The Path_1 usage plot is flat, and high rates does not change it much. The Path_3 usage increases at high rates, with 12% usage at 10000 pps rate.

The CPN network paths usage plots show that all paths are loaded with some portion of overall traffic. Even though Path_4 is not selected by QOS_LOSS, QOS_DELAY does not ignore it by using it in $\approx$ 8%, so that the resulting usage in the Network is about 4%. The Path_1 is selected by cpn002 in 20% of time and shares a second place of usage with Path_2 in cpn002 results, but cpn026 uses Path_1 only in 10% of time. It is not surprising that the Path_1 in the Network is used on average 15% of time and has got the third place.

In conclusion, the experimental results in this section showed that QOS_LOSS,

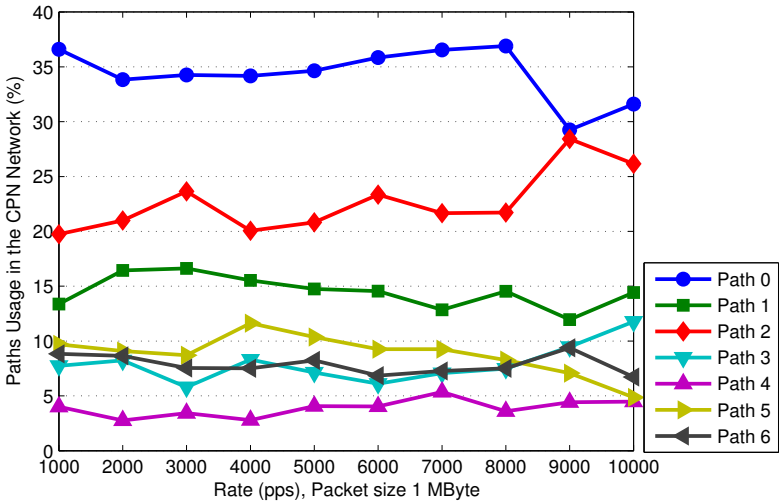

Figure 3.15: The percentage of Paths Usage in the tested CPN network

which is SP Loss QoS in the intermediate nodes and DP Loss QoS with path tracking, manages to maintain DP Loss below 1% for high packet rates, close to the interface bandwidth. The results also proved the effectiveness of the Bilateral traffic differentiation algorithms of the CPN protocol. The four QoS simulation experiment showed that there are 3 or 4 distinct simultaneous paths in the Network in 70% of time. The percentage of time, there is one path for four traffic flows in the Network, is only about 1%. The paths usage plots revealed that the top used paths are the short length paths - Path_0, Path_2 and Path_3. Nevertheless, the longest length Path_6 has never been the less used path and shared the places of usage with medium length paths. QOS_LOSS selects the Path_6 as often as short length Path_1. The paths usage also showed that QOS_LOSS and QOS_DELAY selection processes and logics differ from each other, allowing to utilize CPN network resources in the smart way.

Chapter 4

Conclusions and Future work

This project introduced the idea of Bilateral Traffic Differentiation to the CPN protocol. If Uplink differ significantly in volume from Downlink traffic, each branch could be treated by CPN network in specifically way by means of two distinct QoS. Two QoS used were QOS_DELAY, it Goal is the shortest delay for packets travelling, and QOS_LOSS, it Goal is the less loss of DPs. The existing CPN protocol code was enhanced with several features to implement the main idea. The Round-Robin principle helps SPs to discover both QoS paths simultaneously, as 50% of SPs are sent with QOS_DELAY and as 50% of SPs are sent with QOS_LOSS. The Flow Control File was designed to track the rates of sent and received traffic, so that the Uplink and Downlink volumes could be compared. The two branches comparison is performed by QoS Decision Logic, the larger traffic branch is assigned with QOS_LOSS, smaller traffic branch - with QOS_DELAY. The QOS_LOSS itself consists of SP Loss algorithm, which resides in intermediate nodes, and DP Loss with paths tracking that resides in the sender nodes.

Experiments with two users, one user per sender node, showed the effectiveness of DP Loss algorithm with paths tracking. The 7 available paths are tracked by this logic and the less DP Loss path is always selected. Thus, the CPN Network saturation did not introduce a lot of degradation to the DP Loss performance, the overall DP Loss was less than 1% for rate equal to the interface bandwidth.

Experiments with four users, two users per node, proved the effectiveness of the Bilateral Traffic Differentiation. The test scenario was designed, so that each user in one sender had a distinct QoS for packets transmission from another user in the same sender.

The ideal case for effective CPN network resources utilisation was four simultaneous distinct paths, one path for each user. The results for all rates showed that in more than 70% of time there are at not less than three (3 or 4) distinct paths in the CPN Network. The worst case, which is only one path used by all four users simultaneously, occurred only in 1% of time. The paths usage percentage confirmed that short length paths are preferable by both QoS. Nevertheless, the longest path was used as frequent as medium length paths. Moreover, the longest path had never been the last in the usage percentage ranking. The less used path turned out to be a medium length path.

The future work on the idea, introduced and investigated in this project, could be experiments running on the larger CPN Network with larger number of available paths. It is expected that the performance of DP Loss algorithm for high rates would be improved. The network interfaces bandwidths could be changed, so that the variety of interfaces speeds is present in the network. In such tests the short length paths could be no more leaders in the paths usage. More sender nodes could be introduced to the tests with two users per node. The number of distinct simultaneous paths could be evaluated and expected to increase, so that the CPN network utilisation efficiency would be also increased.

Bibliography

[1] E. Gelenbe, R. Lent, and A. Nunez, "Self-aware networks and QoS," *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1478–1489, September 2004.

[2] E. Gelenbe, G. Sakellari, and M. D. Arienzo, "Admission of qos aware users in a smart network," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 3, no. 1, March 2008.

[3] E. Gelenbe, P. Liu, and J. Laine, "Genetic algorithms for autonomic route discovery," in *Distributed Intelligent Systems: Collective Intelligence and Its Applications, 2006. DIS 2006. IEEE Workshop on.* IEEE, New York, NY, USA, June 2006, pp. 371–376.

[4] E. Gelenbe, R. Lent, A. Montuori, and Z. Xu, "Cognitive packet networks: QoS and performance," in *Proceedings of the IEEE MASCOTS Conference*, Ft. Worth, TX, October 2002, pp. 3–12, opening Keynote Paper.

[5] 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.

[6] E. Gelenbe, M. Gellman, and G. Loukas, "An autonomic approach to denial of service defence," in *Proceedings of the International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM '05).* IEEE, New York, NY, USA, June 2005, pp. 537–541. [Online]. Available: http://dx.doi.org/10.1109/WOWMOM.2005.24

[7] E. Gelenbe and C. Morfopoulou, "A Framework for Energy Aware Routing in Packet Networks," *The Computer Journal*, vol. 54, no. 6, June 2011.

[8] E. Gelenbe and T. Mahmoodi, "Distributed Energy-Aware Routing of Traffic," in *Proceedings of the 26th International Symposium on Computer and Information Sciences (ISCIS'11)*, London, UK, 26-27 September 2011.

[9] R. Lent, "A sensor network to profile the electrical power consumption of computer networks," in *IEEE GLOBECOM 2010 Workshop on Green Computing and Communications (accepted)*, Miami, Florida, USA, 6-10 December 2010.

[10] E. Gelenbe, "Steps toward self-aware networks," *Commun. ACM*, vol. 52, no. 7, pp. 66–75, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1538788.1538809

[11] E. Gelenbe and P. Liu, "QoS and routing in the cognitive packet network," in *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, June 2005, pp. 517–521.

[12] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural Comput.*, vol. 1, no. 4, pp. 502–510, Dec. 1989. [Online]. Available: http://dx.doi.org/10.1162/neco.1989.1.4.502

[13] ——, "Stability of the random neural network model," in *Proceedings of the EURASIP Workshop 1990 on Neural Networks.* London, UK, UK: Springer-Verlag, 1990, pp. 56–68. [Online]. Available: http://dl.acm.org/citation.cfm?id=646655.700113

[14] E. Gelenbe and K. F. Hussain, "Learning in the multiple class random neural network," *Trans. Neur. Netw.*, vol. 13, no. 6, pp. 1257–1267, Nov. 2002. [Online]. Available: http://dx.doi.org/10.1109/TNN.2002.804228

[15] E. Gelenbe and S. Timotheou, "Synchronized interactions in spiked neuronal networks," *Comput. J.*, vol. 51, no. 6, pp. 723–730, Nov. 2008. [Online]. Available: http://dx.doi.org/10.1093/comjnl/bxn004

[16] ——, "Random neural networks with synchronized interactions," *Neural Comput.*, vol. 20, no. 9, pp. 2308–2324, Sep. 2008. [Online]. Available: http://dx.doi.org/10.1162/neco.2008.04-07-509

[17] E. Gelenbe, Z.-W. Mao, and Y.-D. Li, "Function approximation with spiked random networks," *Neural Networks, IEEE Transactions on*, vol. 10, no. 1, pp. 3–9, 1999.

[18] ——, "Approximation by random networks with bounded number of layers," in *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop.*, 1999, pp. 166–175.

[19] E. Gelenbe, "Learning in the recurrent random neural network," *Neural Comput.*, vol. 5, no. 1, pp. 154–164, Jan. 1993. [Online]. Available: http://dx.doi.org/10.1162/neco.1993.5.1.154

[20] E. Gelenbe, Y. Feng, and K. Krishnan, "Neural network methods for volumetric magnetic resonance imaging of the human brain," *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1488–1496, 1996.

[21] E. Gelenbe, M. Sungur, C. Cramer, and P. Gelenbe, "Traffic and video quality with adaptive neural compression," *Multimedia Syst.*, vol. 4, no. 6, pp. 357–369, Dec. 1996. [Online]. Available: http://dx.doi.org/10.1007/s005300050037

[22] J. Aguilar and E. Gelenbe, "Task assignment and transaction clustering heuristics for distributed systems," *Inf. Sci.*, vol. 97, no. 1-2, pp. 199–219, Mar. 1997. [Online]. Available: http://dx.doi.org/10.1016/S0020-0255(96)00178-8

[23] E. Gelenbe, A. Ghanwani, and V. Srinivasan, "Improved neural heuristics for multicast routing," *Selected Areas in Communications, IEEE Journal on*, vol. 15, no. 2, pp. 147–155, 1997.

[24] E. Gelenbe, V. Koubi, and F. Pekergin, "Dynamical random neural network approach to the traveling salesman problem," in *Systems, Man and Cybernetics, 1993. 'Systems Engineering in the Service of Humans', Conference Proceedings., International Conference on*, 1993, pp. 630–635 vol.2.

[25] E. Gelenbe, P. Liu, and J. Laine, "Genetic Algorithms for Route Discovery," *IEEE Transactions on Systems, Man and Cybernetics B*, vol. 36, no. 6, pp. 1247–1254, December 2006.