Measurement and Performance of a Cognitive Packet Network *

Erol Gelenbe, Ricardo Lent, Zhiguang Xu
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816
{erol,rlent,zgxu}@cs.ucf.edu

November 27, 2001

Abstract

As the size of the Internet grows by orders of magnitude both in terms of users, number of IP addresses, and number of routers, and as the links we use (be they wired, optical or wireless) continuously evolve and provide varying reliability and quality of service, the IP based network architecture that we know so well will have to evolve and change. Both scalability and QoS have become key issues. We are currently conducting a research project that revisits the IP routing architecture issues and proposes new designs for routers. As part of this effort, this paper discusses a packet network architecture called a *"Cognitive Packet Network (CPN)"*, in which intelligent capabilities for routing and flow control are moved towards the packets, rather than being concentrated in the nodes. In this paper we outline the design of the Cognitive Packet Network architecture, and discuss the Quality-of-Service based routing algorithm that we have designed and implemented. We then present our test-bed and report on extensive measurement experiments that we have conducted.

Keywords:*Packet based routing, Quality of Service, Reinforcement Learning, Cognitive Packet Networks, Performance Measurement*

1 Introduction

In several recent papers [3, 4, 5] we have proposed a new network architecture called "Cognitive Packet Networks (CPNs)". These are store-and-forward packet networks in which intelligence is built into the packets, rather than at the routers or in the high-level protocols. CPNs carry three major types of packets: smart packets, dumb packets and acknowledgments (ACK). Smart or cognitive packets route themselves, they learn to avoid congestion and to avoid being lost or destroyed. They learn from their own observations about the network and from the experience of other packets. They rely minimally on routers. When a smart packet arrives at a destination, an acknowledgment (ACK) packet is generated by the destination and the ACK heads back to the source of the smart packet along the reverse route. As it traverses successive routers, it is used to update mailboxes in the CPN routers, and when it reaches the source node it provides source routing information for dumb packets. Dumb CPN packets of a specific QoS class use successful routes which have been selected in this manner by the smart packets of that class.

This paper reviews the basic concepts of CPNs. Then, we present analytical results for best and worst case performance. We then describe a test-bed network we have designed and implemented in order to demonstrate these ideas. We provide measurement data on the test-bed to illustrate the capacity of the network to adapt to changes in traffic load and to failures of links. Finally we report measurements which evaluate the impact of the ratio of smart to dumb packets on the end-to-end delay experienced by all of the packets.

*The research reported in this paper was supported by U.S. Army Simulation and Training Command and by Giganet Technologies, Inc.

2 Cognitive Packets and CPNs

Learning algorithms and adaptation have been suggested for telecommunication systems in the past [7, 10]. However these concepts have not been fully exploited in networks because of the lack of an adequate framework allowing decentralized control of communications. In Cognitive Packet Networks, smart packets serve as "explorers" for different source-destination pairs; they rely minimally on routers, so that network nodes only serve as buffers, mailboxes and processors. We use the term Cognitive Packet (CP) and "smart packet" (SP) interchangeably. Smart packets can be grouped into classes so that a group of packets share the same goal, and make use of each other's experience. Upon arrival of a smart packet at its destination, the destination node creates an acknowledgment, which will follow the reverse of the route recorded by the smart packet on its way to the destination. The acknowledgment informs the source about the path that should be followed by dumb packets on their way to this specific destination. Dumb packets are given the path to follow to their destination by the source.

A node in the CPN acts as a storage area for CP's and for mailboxes which are used to exchange data between CPs , and between CP's and the node. It has an input buffer for CPs arriving from the input links, a set of mailboxes, and a set of output buffers which are associated with output links. Nodes in a CPN carry out the following functions:

1. A node receives packets via a finite set of ports and stores them in an input buffer.
2. It transmits packets to other nodes via a set of output buffers. Once a CP is placed in an output buffer, it is transmitted to another

   destination node with some priority indicated in the output buffer.
3. A node receives information from CPs which it stores in Mailboxes (MBs). Mailboxes may be reserved for certain classes of CPs , or may be specialized by classes of CPs . For instance, there may be different MB's for packets identified by different Source-Destination (S-D) pairs.
4. A node executes the code for each CP in the input buffer. During the execution of the CP's code, the CP may ask the node to decline its identity, and to provide information about its local connectivity (i.e., "This is Node A, and I am connected to Nodes B, C, D via output buffers) while executing its code. In some cases, the CP may already have this information in its CM as a result of the initial information it received at its source, and as a result of its own memory of the sequence of moves it has made. As a result of this execution:
   - The CMs of the packets in the input buffer are updated,
   - Certain information is moved from CPs to certain MBs,
   - A CP which has made the decision to be moved to an output buffer is transfered there, with the priority it may have requested.

3 Routing Algorithm using Reinforcement Learning

We use Random Neural Networks (RNN) with reinforcement learning (RNNRL) in order to implement the Smart packet routing algorithm. A recurrent RNN is used both for storing the CM and making decisions. The weights of the RNN are updated so that decisions are reinforced or weakened depending on how they have been observed to contribute to the success of the QoS goal. Our earlier experience with simulations of CPN [5, 12] as well as our current test-bed implementation have shown the practical validity of this choice.

For the reinforcement learning approach to CP adaptation, we have used the RNN [9] which is an analytically tractable spiked random neural network model whose mathematical structure is akin to that of queuing networks. It has "product form" just like many useful queuing network models, although it is based on non linear mathematics. The state q_i of the $i-th$ neuron in the network is the probability that it is excited. The q_i, with $1 \leq i \leq n$ satisfy the following system of non linear equations:

$$q_i = \lambda^+(i)/[r(i) + \lambda^-(i)], \tag{1}$$

where

$$\lambda^+(i) = \sum_j q_j w^+_{ji} + \Lambda_i, \qquad \lambda^-(i) = \sum_j q_j w^-_{ji} + \lambda_i. \tag{2}$$

Here w^+_{ji} is the rate at which neuron j sends "excitation spikes" to neuron i when j is excited, w^-_{ji} is the rate at which neuron j sends "inhibition spikes" to neuron i when j is excited, and $r(i)$ is the total firing rate from the neuron i. For an n neuron network, the network parameters are these n by n "weight matrices" $\mathbf{W}^+ = \{w^+(i,j)\}$ and $\mathbf{W}^- = \{w^-(i,j)\}$ which need to be "learned" from input data. Various techniques for learning may be applied to the RNN. These include Hebbian learning (which will not be discussed here since it is too slow and relatively ineffective with small networks), and Reinforcement Learning which we have implemented for CPN.

There can be different ways to apply Reinforcement Learning in the RNN model. In this paper we have used an approach which was originally suggested for navigation in a maze using an RNN having only excitatory weights, and which we have extended by using an RNN with both excitatory and inhibitory weights. The simulations and implementations over our test-bed have revealed that this simple approach is effective for autonomous routing of the CP's.

Given the Goal G that the CP has to achieve as a function to be minimized (i.e., Transit Delay or Probability of Loss, or a weighted combination of the two), we formulate a reward R which is simply $R = G^{-1}$. Successive measured values of the R are denoted by R_l, $l = 1, 2,$ .. These are first used to compute a decision threshold:

$$T_l = aT_{l-1} + (1-a)R_l, \tag{3}$$

where a is some constant $0 < a < 1$, typically close to 1.

An RNN with as many nodes as the decision outcomes is constructed. Let the neurons be numbered 1, ... ,n. Thus each decision i corresponds to some neuron i. Decisions in this RL algorithm with the RNN are taken by selecting the decision j for which the corresponding neuron is the most excited, i.e., the one with the largest value of q_j. Note that the $l-th$ decision may not contribute directly to the $l-th$ observed reward because of time delays between cause and effect.

Suppose we have now taken the $l-th$ decision which corresponds to neuron j, and that we have measured the $l-th$ reward R_l. Let us denote by r_i the firing rates of the neurons before the update takes place. We first determine whether the most recent value of the reward is larger than the previous "smoothed" value of the reward which we call the threshold T_{l-1}. If that is the case, then we increase very significantly the excitatory weights going into the neuron that was the previous winner (in order to reward it for its new success), and make a small increase of the inhibitory weights leading to other neurons. If the new reward is not better than the previously observed smoothed reward (the threshold), then we simply increase moderately all excitatory weights leading to all neurons, except for the previous winner, and increase significantly the inhibitory weights leading to the previous winning neuron (in order to punish it for not being very successful this time). This is detailed in the algorithm given below. We compute T_{l-1} and then update the network weights as follows for all neurons $i \neq j$:

- If $T_{l-1} \leq R_l$
  - $w^+(i,j) \leftarrow w^+(i,j) + R_l$,
  - $w^-(i,k) \leftarrow w^-(i,k) + \frac{R_l}{n-2}, \; if \; k \neq j.$
- Else
  - $w^+(i,k) \leftarrow w^+(i,k) + \frac{R_l}{n-2}, k \neq j,$
  - $w^-(i,j) \leftarrow w^-(i,j) + R_l.$

Since the relative size of the weights of the RNN, rather than the actual values, determine the state of the neural network, we then re-normalize all the weights by carrying out the following operations. First for each i we compute:

$$r^*_i = \sum_1^n [w^+(i,m) + w^-(i,m)], \tag{4}$$

and then re-normalize the weights with:

$$w^+(i,j) \leftarrow w^+(i,j) * \frac{r_i}{r_i^*},$$
$$w^-(i,j) \leftarrow w^-(i,j) * \frac{r_i}{r_i^*}\ .$$

Finally, the probabilities q_i are computed using the non linear iterations (1), (2), leading to a new decision to send the packet forward to the output link which corresponds to the neuron which has the largest excitation probability.

4 An Analytical Model of Smart Packets in CPN

We have examined a simple mathematical model to predict the end-to-end delay experienced by smart packets. The worst case performance is obtained by considering SPs which try to find the route to their destination by moving at random, with two constraints related to the topology of the network which will be discussed below. The network topology that we use to evaluate the worst case is identical to the one which we have used in our simulations, except that it has a cylindrical topology with "R" circles or rows, each containing "a" nodes, making up the cylinder. There are no left or right "edge" nodes in the network used in the analytical model. Each node on the two (top and bottom) "end" circles serves both as a source and as a destination for packets. The two routing constraints for packets, which we mentioned above are:

- A SP originating at the top row never heads upwards, and if it originates at the bottom row it never heads downwards.
- A SP which is one link away from its destination will directly go to its destination without any further random search, because packet networks the outgoing link of a node will carry the identity of the node which is at the other end of the link.

Our analysis provides the following results under the assumption that packets at any source may head to any destination, and that smart packets do not know the direction they should head in, except that they need to go from the top row to the bottom row (or vice versa).

- Average Number of Nodes Visited in Row i by a Smart Packet When There Are No Losses

$$A(i) = \begin{cases} \frac{1}{f_i} & i \leq i \leq R-2 \\ \frac{a}{(1+(a-1)f_i)} & i = R-1 \\ \frac{a}{2} & i = R \end{cases}$$

  where f_i is the probability that a smart packet leaves a node in row i to move to the next row.

- Average Number of Nodes Visited in Row i by a Smart Packet with Losses at each Node

$$A_l(i) = \begin{cases} \frac{1}{l_i+f_i(1-l_i)} & i \leq i \leq R-2 \\ \frac{1}{1-(1-f_i)(1-l_i)(1-\frac{1}{a})} & i = R-1 \\ \frac{a}{2+(a-2)l_i} & i = R \end{cases}$$

  where l_i is the probability that a loss can occur at some node in row i of the network.

- The Probability that a Smart Packet Is Lost as It Traverses Row i

$$\pi(i) = 1 - \frac{A_l(i)}{A(i)},\ 1 \leq i \leq R$$

- The Probability that a smart packet eventually enters row i

$$P_e(i) = \begin{cases} 1 & i = 1 \\ (1-\pi(i-1)) \cdot P_e(i-1) & 2 \leq i \leq R \end{cases}$$

- The Average Number of Times that a Randomly Selected Smart Packet Visits a Node In the row i

$$e_l(i) = P_e(i) \cdot \frac{A_l(i)}{a}, \quad 1 \leq i \leq R$$

under the assumption that the traffic in the network is homogeneous.

- The Probability that a Smart Packet Is Lost as It Traverses the Network

$$P_l^s = 1 - \Pi_{i=1}^{R}(1 - \pi(i))$$

- The Effective Traffic from S to D

$$\lambda(S, D) = \frac{\lambda^0(S, D)}{1 - P_l^s}$$

where $\lambda^0(S, D)$ is the offered S to D traffic.

- The Average Traffic of Smart Packets Entering a Node in Row i
  - For unilateral traffic going just from the top to the bottom of the network

$$\lambda_s(i) = \sum_S \sum_D e_l(i) \cdot \lambda(S, D)$$

  - For symmetric bilateral traffic going both from top to bottom and vice versa

$$\lambda_s(i) = \sum_S \sum_D (e_l(i) + e_l(R - i + 1)) \cdot \lambda(S, D)$$

- The Average Number of Smart Packets at a Node in Row i

$$R(i) = \frac{\lambda_s(i)}{\gamma - \lambda_s(i)}$$

where γ is the average service rate at each Node.

- The Overall Average Delay of Smart Packets
  - For unilateral traffic going just from the top to the bottom of the network

$$r = \frac{\sum_i a \cdot R(i)}{\sum_S \sum_D \lambda^o(S, D)}$$

  - For symmetric bilateral traffic going both from top to bottom and vice versa

$$r = \frac{\sum_i a \cdot R(i)}{\sum_S \sum_D \lambda^o(S, D) \cdot 2}$$

To illustrate these results, we have varied f_i, the probability a smart packet leaves a node in row i to move to the next row. The numerical results of the leftmost graph in Figure 1 show that the larger the value of f_i, the smarter the packets are, and consequently, the smaller the average response time R. The *best case* performance can be achieved if the following three conditions are satisfied:

- The traffic load is evenly distributed among all the nodes in the network;
- Each packet takes the shortest path from its source to its destination under the assumption that the number of nodes visited by a packet is the dominant factor among those that determine its delay;

- There is no packet loss.

Take an arbitrary packet, let "s" and "d" represents its source and destination respectively. The length of the shortest path that it can possibly take is $M = |d - s| + R$ with expected value $\overline{M} = \frac{a}{2} + R$. Since all nodes are equally loaded, the packet arrival rate to each node is

$$\lambda_n = \frac{2a\lambda^0(S,D)\overline{M}}{aR} = (\frac{a}{R} + 2)\lambda^0(S,D)$$

where $\lambda^0(S,D)$ is the offered S to D traffic. Similar to the previous worst case analysis, now we can adopt the queuing theory to obtain the best case average packet delay:

$$r = \frac{aR\frac{\lambda_n}{\gamma - \lambda_n}}{2a\lambda^0(S,D)} = \frac{\frac{a}{2} + R}{\gamma - (\frac{a}{R} + 2)\lambda^0(S,D)}$$

The following figures conclude our smart packet analysis. The one on the left shows the best case average packet delay and the one on the right illustrates the performance comparison among the best case, the worst case and the simulation (RNN with reinforcement learning) in terms of the average packet delay.

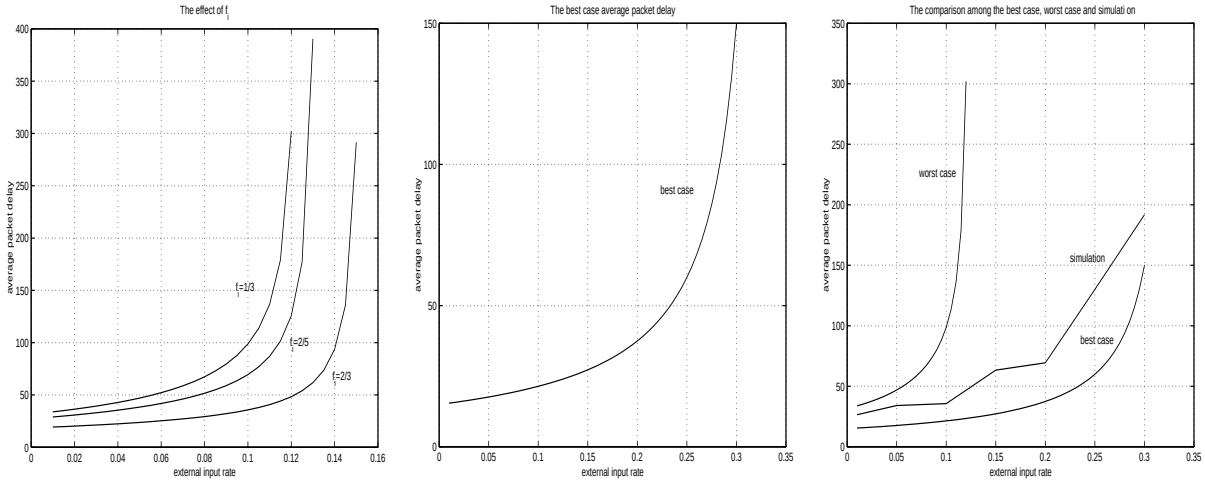

Figure 1: Comparison between the Worst Case (left), Best Case (center), and the Analytical Worst and Best Cases Compared to RNN-RL Learning Based Simulation (right)

5 An experimental CPN test-bed: protocol design and software implementation

In this section we describe aspects of the CPN protocol design and details of a Test-bed implementation. The software has been integrated into the Linux kernel 2.2.x. with minimal changes in the existing networking code and is independent of the physical transport technology. The network interface is compatible with the popular BSD4.3 socket layer in Linux, and provides a single application program interface (API) for the programmer to access the CPN protocol. The Linux kernel support for low cost PCs and a growing number of platforms, and the freely availability of its source code, makes Linux an attractive system for the development of a project of this nature.

A CPN provides a connectionless service to the application layer, and consists of a set of hosts interconnected by links of some kind, where each host can operate both as an end node of communication and/or as a router. The addressing scheme utilizes a single number of 32 bits to represent the CPN address of each node.

Cognitive Packets (CP) use the network to transport user data or routing information. We have considered three types of CP's addressing particular goals: Smart packets (SP), Dumb packets (DP) and Acknowledgment packets (AP). SPs and DPs transport user data whereas APs transport routing and delay information. The SPs principal goal is to find the best route to a given destination. SPs are sent to the network with the objective of discovering new destinations for the source, update existing information, or seek better routes to known destinations. DPs use the information acquired by SPs to transport user data using the best possible route. The current version of the test-bed is based on PCs with Linux as the Operating System.

We are experimenting with various topologies where each CPN router can be connected up to four other routers and Figure 2 depicts a typical topology. All the nodes have been configured to use CPN and IP packets at the same time for comparison purposes. CP's are of variable size and consist basically of three areas: a header, a Cognitive map (CM) and a data portion. Each port of a CPN node uses a 10 Mbps Ethernet link connected with another CPN node. The physical connection between routers uses a crossover twisted pair copper cable.

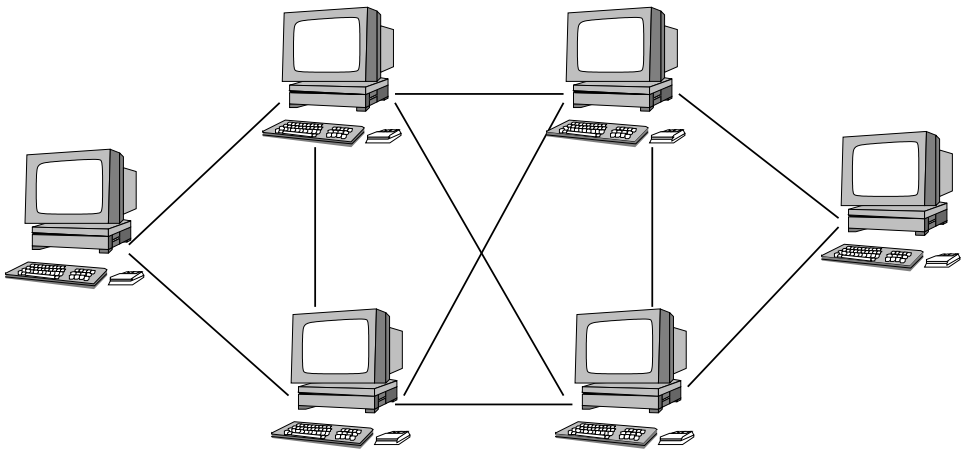

Figure 2: The current test-bed topology

6 Network Measurements

The purpose of the measurements we describe on the test-bed is to evaluate the CPN architecture with respect to its dynamic behavior and ability to adapt to changing network conditions. These conditions include both changes in traffic patterns and possible failures in links. With respect to the network topology described in Figure 2, we have conducted measurements for a main flow of traffic from node Node 10 (left) to Node 5 (right). For the first set of experiments, the input rate was fixed to 5 packets second. In several of the experiments we have conducted there is a flow of "obstructing" traffic over and above the main traffic flow from Node 10 to Node 5.

Each time we have added this obstructing traffic it has been on one or more specific links, and at a relatively high traffic rate of 5700 packets per second in each direction (i.e., bi-directional on each selected link).

To simplify matters we have selected the length of the payload for all packets to be of fixed length of 1000 bytes (B). Additionally, there is a CPN header (20 bytes) for each packet. Since we use Ethernet at the transport level we need to add an Ethernet header and tail to each packet. For dumb packets we also have the source route information which acts as the cognitive memory for a total of $16(N-1)$ bytes, where N is the number of nodes in the path. Thus the total packet length is in excess of 1100 bytes. The resulting obstructing traffic is therefore in excess of $5.7MB/sec$ per obstructed link, in each direction.

The $x-axis$ in all the plots refers to successive packets and is scaled in packet counts. In each plot the $y-axis$ presents delay in milliseconds (left) or route number (right). Note that all time values are

rounded up to the closest integer number of milliseconds, while the route numberings are indicated in the figure captions. On the plots, an "X" under the $x - axis$ in the route plot indicates an instance of packet loss of either smart or dumb packets.

Figure 3 shows a traffic pattern with 20% of SPs and 80% of DPs. Here the network is only carrying the traffic from Node 10 to Node 5 with no interfering traffic. The successive individual delay values and individual routes taken by close to 200 packets is traced packet by packet. Each route label corresponds to some particular sequence of nodes which are traversed.

We observe that routes do change despite the fact that there is no traffic on the network other than the one that is being traced. This is a consequence of sending out a constant proportion of SPs to test alternate routes, resulting in the selection of a new outgoing link at a node if the delay is estimated to be smaller according to the CPN algorithm.

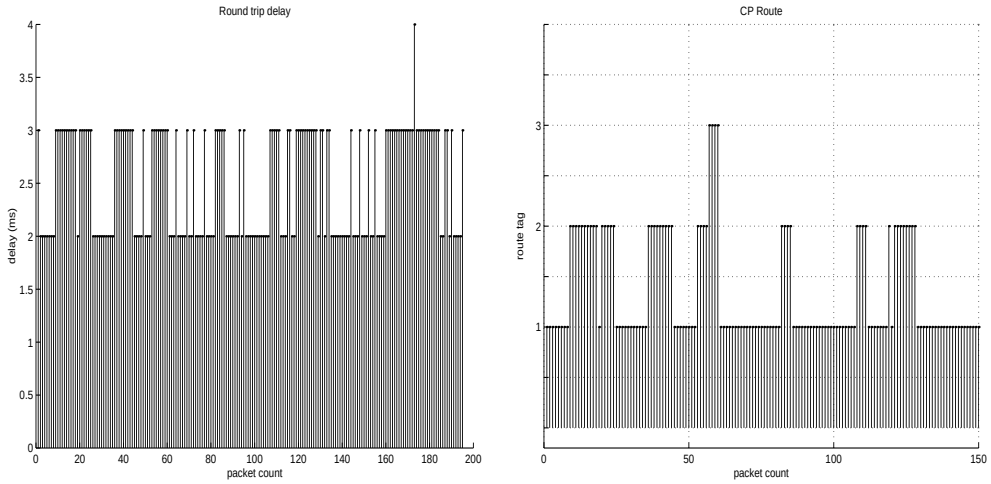

Figure 3: Network with no obstructing traffic.

In Figure 4 we plot the delays and routes experienced by individual packets on the traffic flow from Node 10 to Node 5. We now have obstructing traffic which perturbs this traffic flow, and we still have 20% of SPs in all traffic flows. The additional traffic is introduced when the packet count reaches 30 on the traffic from Node 10 to Node 5. The obstructing traffic flows on the link from Node 10 to Node 1 . We see that when the obstructing traffic is initialized, the main traffic flow encounters significant delays as well as losses. Then, thanks to the CPN algorithm, the network determines a new route and delays go back to a low level. Further spurious increases in delay occur but are short-lived each time the SPs probe the network for better routes and some losses do occur again.

The network's reaction to link failures was tested and is reported in Figure 5. The network has no obstructing traffic during this experiment so that it only carries traffic from Node 10 to Node 5 with a fixed 20% ratio of SPs. At packet count 40, the link from Node 10 to Node 1 was disconnected and we immediately observed the loss of a few packets. The initially used preferential Route 1 of Nodes 10:1:4:5 was rapidly abandoned by the CPN algorithm, and the system switched to Route 2 which uses Nodes 10:2:1:4:5 = 2. Due to continued searching by SPs, later Route 3 (Nodes 10:2:3:5) and Route 2 are used, but Route 1 is not used again.

In the next set of results, we show the effectiveness of the CPN to adapt to different quality of service requirements, for this, we have introduced jitter control as a new component for the routing decision of the smart packets in addition to the average delay. Along with the main flow of packets we introduce a second flow to produce obstructing traffic to the former. The resulting average delay after the transmission of 200 thousand packets is shown in Figure 6. The x-axis represent the inter-packet time for the main flow of packets. There is a significant improvement in the overall standard deviation of the packets when the jitter control is active as shown in Figure 7.

Another important aspect that we investigate is the effect of the ratio of SPs on overall performance.

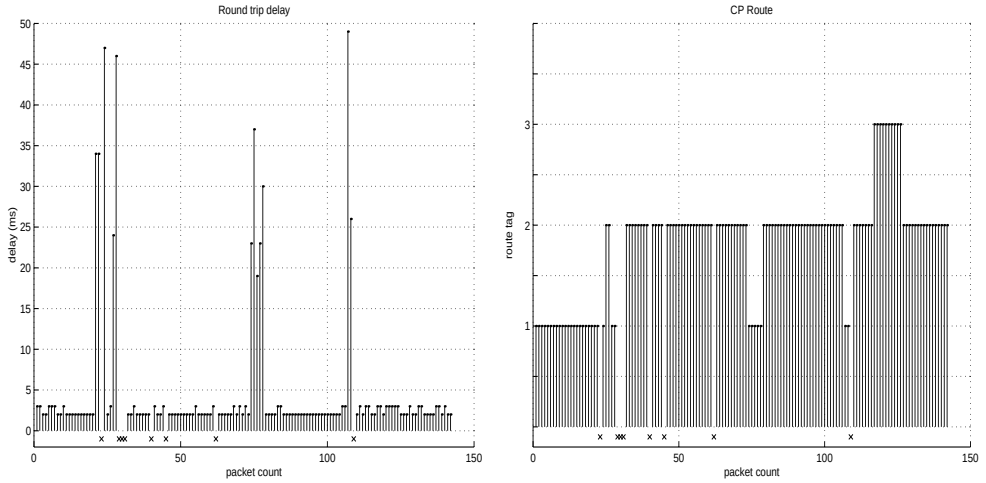

Figure 4: Network with obstructing traffic.

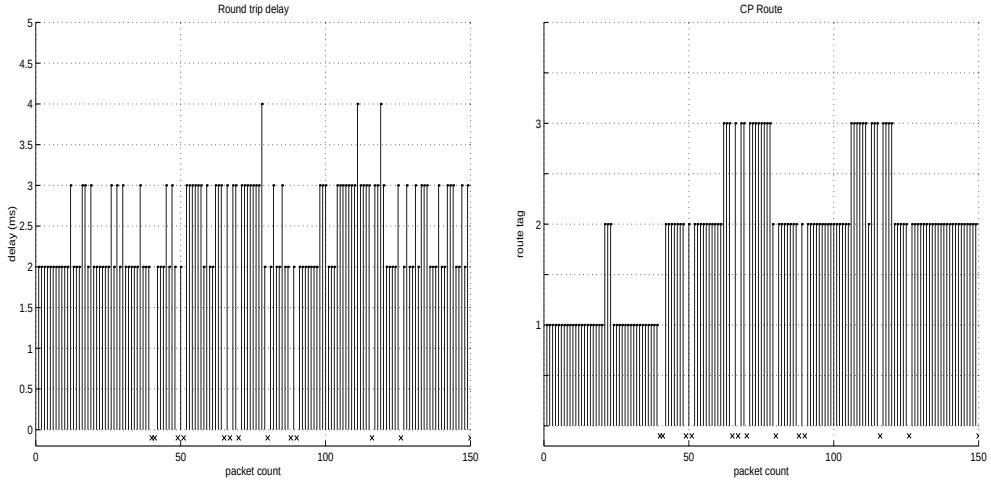

Figure 5: Measurements concerning the network under link failure

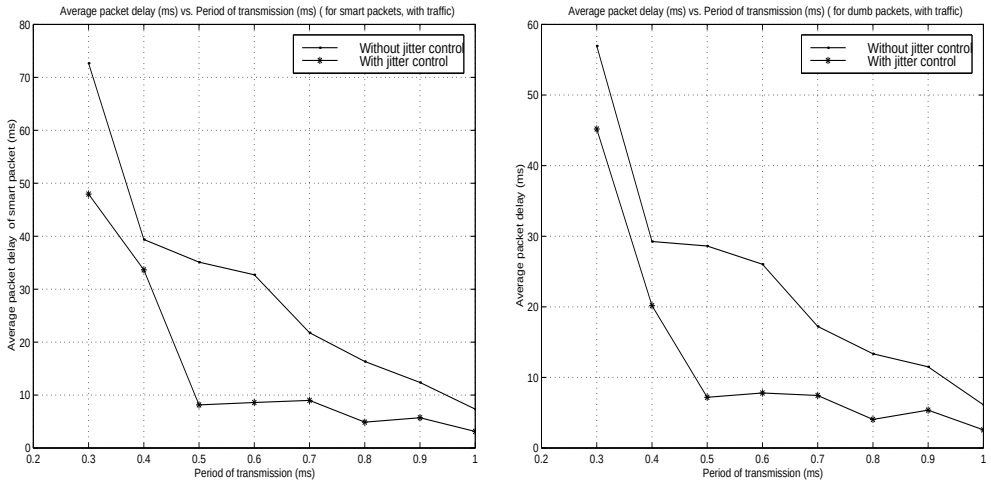

Figure 6: Average round-trip delay for smart (left) and dumb (right) packets under regular routing conditions and routing with jitter control

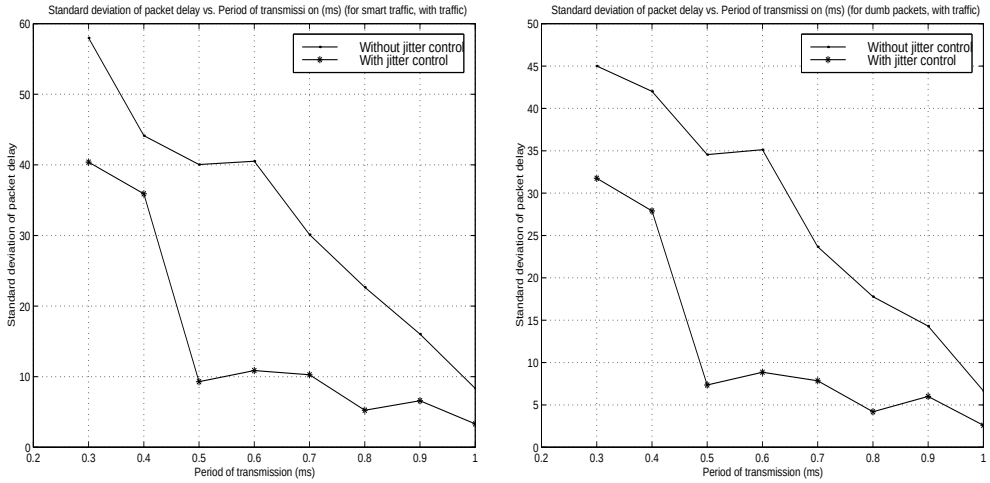

Figure 7: Standard deviation of delay for smart (left) and dumb (right) packets under regular routing conditions and routing with jitter control

We have conducted a set of experiments using a fixed input rate of 2500 packets per second to examine this point and our results are reported in Figure 8. The interesting result we observe is that as far as the DPs are concerned, when we have 15% or 20% of SPs we have achieved the major gain in delay and jitter reduction. Going beyond those values does not significantly reduce the delay for DPs.

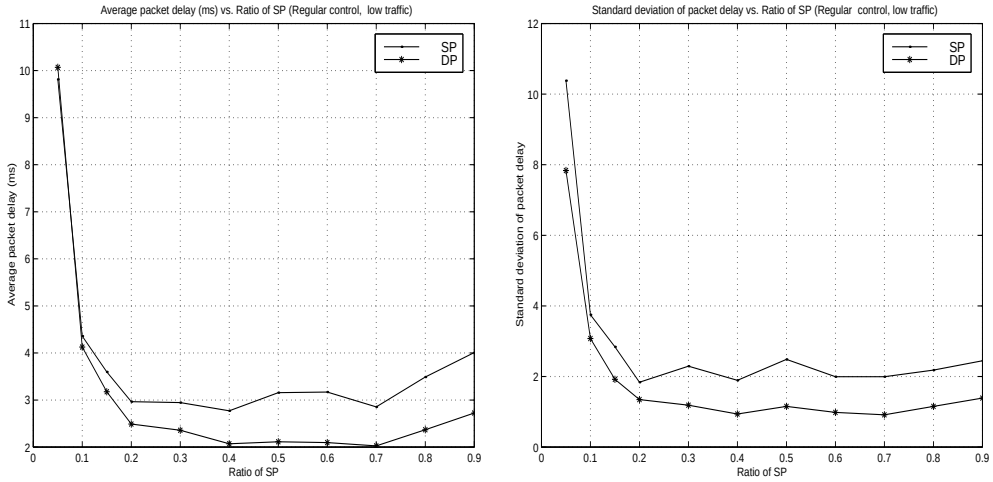

Figure 8: Average round-trip delay (left) and standard deviation of delay for smart (left) and dumb packets as a function of the percentage of smart packets

7 Conclusions

Cognitive Packet Networks (CPN) are a new packet network paradigm which address some of the needs of global networking.

CPN simplifies router architecture by transfering the control of QoS based best-effort routing to the packets, away from the routers. Routing tables are replaced by reinforcement-algorithm based routing functions. A CPN carries three distinct types of packets: Smart or cognitive packets which search for routes based on a QoS driven reinforcement learning algorithm, ACK packets which bring back route

information and measurement data from successful smart packets, and Dumb packets which do source routing.

In this paper we have summarized the basic principles of CPNs. Then we have described the Reinforcement Learning (RL) algorithm which taylors the specific routing algorithm to the QoS needs of a class of packets. In order to evaluate the effectiveness of smart packets, we have derived analytical results for their best and worst-case performance. We have then described in some detail the design and implementation of our current test-bed network which uses ordinary PC-based workstations as routers.

Interconnection between routers makes use of 10Mb/s point-to-point Ethernet connections. Although the current test-bed is limited to six routers, the design and implementation is general and can be directly extended to an arbitrary number of routers. The CPN software has been implemented in a Linux environment and is portable to a wide range of platforms.

Finally we have provided extensive measurement data on the test-bed to illustrate the capacity of the network to adapt to changes in traffic load and to failures of links. Some of the measurements we report present traces of short sequences (under 200 packets) of packet transmissions showing individual packet delays, paths taken by the packets and individual packet losses due to link failures or buffer overflows.

Other measurements we report are based on long-term measurements covering hundreds of thousands of packet transmissions. These long term measurements report average packet delays for Smart and Dumb packets with and without obstructing traffic. They also show how different QoS goals (e.g., Delay or Jitter) impact performance. We also measure packet loss rates, rates of packet desequencing, Measurements are also reported to evaluate the impact of the ratio of Smart packets to total packets, on the end-to-end delay experienced by all of the packets. We see that a 10-20% ratio of Smart packets is best for the performance of Dumb packets. In heavy traffic conditions, if the ratio of Smart packets becomes too high, the Dumb packet delay can become very significant. Ongoing work in this project includes the deployment of a large test-bed, the inclusion of wireless links, and the design of single-card routers leading to very low cost routing technologies which use CPN.

8 Vitae

Erol Gelenbe is the author of several books in the area of queuing networks and computer system performance which have appeared in English, French, Japanese and Korean. He has graduated some 55 PhD students and published over 100 articles in leading journals of Computer Science, Electrical Engineering and applied Probability. He currently works on novel network architectures and protocols, as well as on multimedia applications. His recent papers have appeared in the *Proceedings of the IEEE*, the *IEEE Journal on Selected Areas in Communications*, the *IEEE Transactions on Neural Networks* and the journal *Performance Evaluation*. He holds the University Chair of Electrical Engineering and Computer Science (EECS) at the University of Central Florida, where he is Director of the School of EECS and Associate Dean of Engineering. He holds a PhD from Polytechnic University (Brooklyn, New York), and the Doctor of Science degree from the University Pierre et Marie Curie (Paris VI) in France. His honors include the "Grand Prix France Telecom" of the French Academy, the IFIP Silver Core, the Officier de l'Ordre National du Mérite of France, an Honorary Doctorate from the University of Rome II, and Fellow of the ACm and of the IEEE. His research is funded by NSF, the US Army and industry.

Ricardo Lent, was born in Chiclayo, Peru. He received the B.S. degree and the "Ingeniero Electronico" title from Universidad Ricardo Palma, Lima, Peru, in 1992, and the M.S. degree in Electrical Engineering from the Universidad Nacional de Ingenieria, Lima, Peru in 1997. From 1992 to 1999 he made major contributions toward the establishment of the first Internet network in Peru. He has also been a Lecturer at the Universidad Nacional de Ingenieria and at the Universidad Nacional Pedro Ruiz Gallo in Peru. He has co-authored several papers in IEEE and other international conferences and is also co-author of a paper in the journal *Performance Evaluation*. He is currently pursuing a doctoral degree in Computer Science at the University of Central Florida, Orlando, FL.

Zhiguang Xu received the undergraduate degree in Computer Engineering from Beijing University of Posts and Telecommunications, China and the M.S. degree in Computer Science from University of

Central Florida. Since January 1999, he has been a Graduate Student working towards the Ph.D. degree in Computer Science at the University of Central Florida. He was with Nortel Networks for four years. His research covers neural-network applications in modelling and simulation and in the design of novel packet switching architectures. He recently co-authored several conference papers, which have appeared in the journals *Proceedings of the IEEE* and *Performance Evaluation.* the *IEEE Conference on Tools for Artificial Intelligence* in 1999, the *International Symposium on Computer and Information Sciences* in 1999, and the *IEEE MASCOTS Workshop 2000* in San Francisco.

References

[1] E. Gelenbe "Probabilistic automata with structural restrictions", *Proc. SWAT 1969 (IEEE Symp. on Switching and Automata Theory)*, also appeared as "On languages defined by linear probabilistic automata", *Information and Control*, Vol. 18, February 1971.

[2] E. Gelenbe "A realizable model for stochastic sequential machines", *IEEE Trans. Computers*, Vol. C-20, No. 2, pp. 199-204, February 1971.

[3] E. Gelenbe, E. Seref, Z. Xu "Towards networks with intelligent packets", *Proc. IEEE-ICTAI Conference on Tools for Artificial Intelligence*, Chicago, November 9-11, 1999.

[4] E. Gelenbe, R. Lent, Z. Xu "Towards networks with cognitive packets," Opening Key-Note Paper, *Proc. IEEE MASCOTS Conference*, ISBN 0-7695-0728-X, pp. 3-12, San Francisco, CA, Aug. 29-Sep. 1, 2000.

[5] E. Gelenbe, R. Lent, Z. Xu, "Towards networks with cognitive packets", Opening Invited Paper, *International Conference on Performance and QoS of Next Generation Networking*, Nagoya, Japan, November 2000, in K. Goto, T. Hasegawa, H. Takagi and Y. Takahashi (eds), "Performance and QoS of next Generation Networking", Springer Verlag, London, 2001.

[6] R. Viswanathan and K.S. Narendra "Comparison of expedient and optimal reinforcement schemes for learning systems", *J. Cybernetics*, Vol. 2, pp 21-37, 1972.

[7] K.S. Narendra and P. Mars, "The use of learning algorithms in telephone traffic routing - a methodology", *Automatica*, Vol. 19, pp. 495-502, 1983.

[8] R.S. Sutton "Learning to predict the methods of temporal difference", *Machine Learning*, Vol. 3, pp. 9-44, 1988.

[9] E. Gelenbe (1993) "Learning in the recurrent random neural network", *Neural Computation*, Vol. 5, No. 1, pp. 154-164, 1993.

[10] P. Mars, J.R. Chen, and R. Nambiar, "Learning Algorithms: Theory and Applications in Signal Processing, Control and Communications", CRC Press, Boca Raton, 1996.

[11] E. Gelenbe, Zhi-Hong Mao, Y. Da-Li (1999) "Function approximation with spiked random networks" *IEEE Trans. on Neural Networks*, Vol. 10, No. 1, pp. 3–9, 1999.

[12] E. Gelenbe, R. Lent, Z. Xu, "Design and performance of Cognitive Packet Networks", to appear in *Performance Evaluation.*