Adaptive Resilience of the Cognitive Packet Network in the Presence of Network Worms

Georgia Sakellari and Erol Gelenbe

Intelligent Systems and Networks Group
Dept. of Electrical & Electronic Engineering
Imperial College London
{g.sakellari, e.gelenbe}@imperial.ac.uk

Abstract. Network worms are malicious self-replicating applications that propagate in a network by infecting node after node. They can cause significant damage by reducing the system performance or totaly disabling nodes, which results into considerable degradation of the quality of service (QoS) that the users experience. The Cognitive Packet Network (CPN) is an adaptive routing protocol that attempts to address the QoS requirements of the network's users by routing their traffic according to them. Although it is generally very resilient to network changes, it may suffer worse performance during a worm attack. Here we evaluate its performance in such crises and compare it with the Open Shortest Path First (OSPF) routing protocol, the current industry standard and widely used in Internet Protocol networks. Then we improve it by introducing a failure detection element that reduces packet loss and delay during failures. Our experiments were performed in a large networking testbed.

1 Introduction

As people and organisations increasingly depend on computer networks, threats such as computer worms gain more attention. These small self-replicating and self-propagating malicious applications exploit system vulnerabilities of some operating systems and spread through networks. Their defining characteristic is their ability to achieve high infection rates; they can spread and saturate a network very quickly. The results of such attacks could be mild, such as a printout of a message or more serious such as deleting or modifying system files, reducing the system performance, or causing total failure to the infected machines. From the service quality perspective and according to the extent of the spread, the latter could lead to serious agitation for the users of the network, due to information loss and delays.

The need for network stability and reliability has led to the growth of autonomic networks that use QoS driven approaches to provide more stable and more reliable communications. A particularly promising such architecture is the Cognitive Packet Network (CPN) that was introduced in [1] and has been shown to adapt quickly to varying network conditions and user requirements. Contrary

to conventional mechanisms, it is the users rather than the nodes that control the routing, by specifying their desired QoS criteria and the network tries to route each one of them individually based on his/her needs. CPN has been evaluated extensively under normal operating conditions and has proven to be very adaptive to network changes such as congestion and link failures. Here we investigate the performance of CPN under catastrophic node failures caused by the spread of Internet worms.

The paper is organised as follows: Section 2 provides a brief summary of the operation of CPN and the learning algorithms it uses for the routing decisions that optimise user-specified QoS goals. In Section 3, we go through the related work on performance evaluation of CPN. In Section 4 we present experiments we conducted specifically for network node failures propagated as Internet worms. In Section 5, we introduce a failure detection element in the CPN mechanism and achieve further improvement in its performance. We conclude in Section 6 with a summary of our contributions and suggested future work.

2 Overview of CPN

CPN is an adaptive packet routing protocol that addresses QoS by using adaptive techniques based on on-line measurements [2–6]. It is a distributed protocol with which users, or the network itself, declare their QoS Goals, such as minimum Delay, maximum Bandwidth, minimum Packet Loss, minimum Variance of the packet delay, maximum Security Level in a path, minimum Power Consumption in a wireless node, or a weighted combination of these. It is designed to perform self-improvement by learning from the experience of special packets that constantly probe the network.

More specifically, it makes use of three types of packets:

- smart packets (SP) for discovery,
- source routed dumb packets (DP) to carry the payload,
- and acknowledgement (ACK) packets to bring back information that has been discovered by SPs, and is used in nodes to train neural networks and produce routing decisions.

The role of SPs is to explore the network and discover the best routes, according to a QoS goal, for each source-destination pair in the network. At each hop SPs are routed according to the experiences of previous packets with the same goals and the same destination. The term “goal” is used instead of “QoS specifications” to emphasize the fact that there are no QoS guarantees and that CPN provides a best effort service [7]. The decisions of the SPs are based on a learning algorithm. In order to explore all possible routes, at some hops, each SP makes a random routing decision, with a small probability (usually 5%). To avoid overburdening the system with unsuccessful requests or packets which are in effect lost, all packets have a life-time constraint based on the number of nodes they have visited.

Several algorithms have been used in CPN as learning and decision techniques in order for SPs to find satisfactory routes from source to destination based on the desired goals. The simplest algorithm used is the Bang-Bang algorithm [8] but the main disadvantage of it is the fact that it uses a priori information. Therefore, other, more sophisticated algorithms were used based on Random Neural Networks. The Random Neural Network (RNN) [9] is a biologically inspired neural network model which is characterised by the existence of positive (excitation) and negative (inhibition) signals in the form of spikes of unit amplitude that circulate among nodes and alter the potential of the neurons. Each neuron can be connected to another neuron and each connection is characterized by an excitatory or inhibitory weight [8]. The state of a neuron, which represents the probability that the neuron is excited, has been proven to satisfy a system of nonlinear equations with a unique solution. Therefore, in a CPN network, at each node a specific RNN that has as many neurons as the possible outgoing links, could represents the decision to choose a given output link for a smart packet. The arrival of SPs triggers the execution of RNN and the routing decision is the output link corresponding to the most excited neuron.

As far as the learning process used with RNN, several learning techniques have been proposed. Hebbian learning was tested at the early stages of the CPN development and was shown to be inefficient and slow [7]. Other algorithms include feedforward learning RNN with the use of a gradient descent quadratic error function. This algorithm is not very computationally efficient because it requires computation at every step and also because mathematical analysis of the model leads to a "back-propagation type algorithm" that requires the solution of a linear and a nonlinear system of equations each time. The algorithm that eventually prevailed in the implementations of CPN is Reinforcement Learning (RL). RL is used to change neuron weights in order to reward or punish a neuron according to the level of goal satisfaction measured on the corresponding output. Therefore the decisional weights of a RNN are increased or decreased based on the observed success or failure of subsequent SPs to achieve the goal. Thus RL will tend to prefer better routing schemes, more reliable access paths and better QoS.

Finally, Genetic Algorithms (GA) have recently been tested in CPN. The authors of [5, 10] use GAs to modify, filter and combine the paths already found by the SPs in order to generate new undiscovered but valid source - destination paths and select the most advantageous ones. Experimental results in [5] have showed that the GA daemon significantly improves QoS under light network traffic but not under high traffic conditions. An explanation given by the authors is that the GA tends to delay decision making, since it stores more information and makes recommendations based on longer term trends.

3 Related work on performance evaluation of CPN

The performance of the CPN routing protocol has extensively investigated in the past but it has not been tested sufficiently in the presence of node failures.

More specifically, CPN's ability to adapt to changing network conditions, such as changes in traffic load, link failures, or buffer overflows has been experimentally evaluated in [11]. The experiments showed that CPN managed to find new routes in order to avoid obstructing traffic introduced in some of the links and also avoided links that were under failures. Another issue studied experimentally in [11] was the effect of ration of SPs on overall performance. The experiments concluded that in order to achieve the best performance for the data packets (DPs) the percentage of SPs that should be send for discovery is 10% to 20% of the data packets' rate. Going beyond these values does not significantly improve the QoS values for DPs.

The authors of [12], also provide experimental, as well as simulation, results to investigate the amount of SPs needed in order CPN to perform well. The results show that a relatively small fraction of SPs and ACKs, compared to total user traffic, is needed to serve the users' QoS Goals. Additionally they show that a small number of SPs can suffice to initially establish a connection.

A set of experiments which demonstrate how CPN performs in a realistic environment of a 46-node testbed have been presented in [13]. The experiments were conducted on a 46-node testbed, also used in our experiments presented in the following Sections, the topology of which represents a real-world topology, the Swiss Education and Research Network (SWITCHlan). The administrators of this network provided the authors of [13] with details on their 46-router backbone, complete with bandwidth, OSPF costs, and link-level delays. CPN's performance under normal operation was compared to that of the Open Shortest Path First (OSPF) routing protocol used in IP networks. The experiments show that the routes CPN computes are as good as those computed a priori using administrator-defined costs. Furthermore, the paper gives experimental results showing that RNN with RL can autonomously learn the best route in the network simply through exploration in a very short time-frame and demonstrates that the CPN protocol is able to adapt to changes in the network environment quickly, by switching to a new optimal route in the network.

Although oscillations are generally considered as a weakness of a network, performance evaluations in [14, 15] indicated that routing oscillations in CPN do not severely degrade performance as would be expected, and high performance can still be obtained even in the presence of oscillations.

The choice of a "goal" and "reward" function for packetized voice applications is discussed in [4] and experiments conducted for "voice over CPN" are presented. The CPN's performance is detailed via several measurements, and the resulting QoS is compared with that of the IP routing protocol under identical conditions showing the gain resulting from the use of CPN.

The experiments in [16] compare a CPN routing where the QoS goal is the minimum hop count with a CPN routing using minimum-delay and a version where routing is based on a combination of hop count and forward delay. The experiments were conducted under low, medium and high background traffic and show that the use of criteria more complex than the shortest number of hops, can provide better overall quality of service.

Measurements indicating how the CPN protocol can respond to different QoS goals are also presented in [5, 7]. Composite goal functions for taken into account both delay and packet loss are proposed. In [7], the measurements suggest that CPN networks effectively adapt routing behavior to the QoS goal that is specified.

The authors of [17] have implemented a composite QoS goal metric which consists of path length and buffer occupancy of nodes to achieve traffic balancing and to identify low-delay paths in a network. Experimental results in a wired testbed and wireless ad hoc simulations show that a routing goal that combines path length and buffer occupancy in nodes offers the advantage of producing approximately the same performance as that of using delay but with a smaller packet overhead.

4 Performance of CPN in the presence of Network Worms

In order to investigate the performance of CPN in the presence of network worms we have developed a failure emulation mechanism. This is based on disabling the Ethernet interfaces of a node which are connected to the network, so that no traffic will be able to go through that node, just like in a real breakdown of a machine. Normal operation of the machine is restored by re-enabling the corresponding Ethernet interfaces.

We have also developed a mechanism with which the failures can be propagated within the network either randomly or according to a distribution model or pattern. This emulator was first presented in the demo session of INFOCOM'08 [18]. In that demo the failure spread was modelled according to the Analytical Active Worm Propagation model [19] where each infected node tries to infect others. The worm spreading of this model depends on many parameters such as the scanning rate, which is the average number of machines scanned by an infected machine per unit time, the patching rate, which is the number of machines that are being patched per unit time, the time a node has to wait before it starts immunising others, and the time a node has to wait before it starts infecting others.

In the experiments presented in this paper the failures are also propagated as a computer worm, spreading around the network and trying to infect it but, in order to reduce randomness and have more clear results, we have chosen a simpler model. More specifically, a node can be in one of the following states: infected, immunised, or vulnerable. The infections are spread randomly around the network according to two parameters, the "scanning rate" and the "failure duration". By *scanning rate* we mean the average number of machines scanned per unit time. So, at each scan one or more node are chosen and if they are immune nothing happens, if they are already infected they are not re-infected (they do not change their infection behaviour) and if they are vulnerable they become infected. Therefore a scan might not always result to an infection (failure). When a node is infected it is considered under failure, traffic cannot go though

it since the Ethernet interfaces are disconnected. The time an infected node will stay under failure is the *failure duration*. After this the node is restored, and the node goes back to normal operation. In order to capture the patching impact on the worm propagation after each failure restoration the node is immunised and cannot be infected again. The scanning mechanism used currently is a random scanning mechanism which is not affected by the number of infected nodes in the network (a node outside the network is the cause of the infection in the network).

4.1 Configuration of the experiments

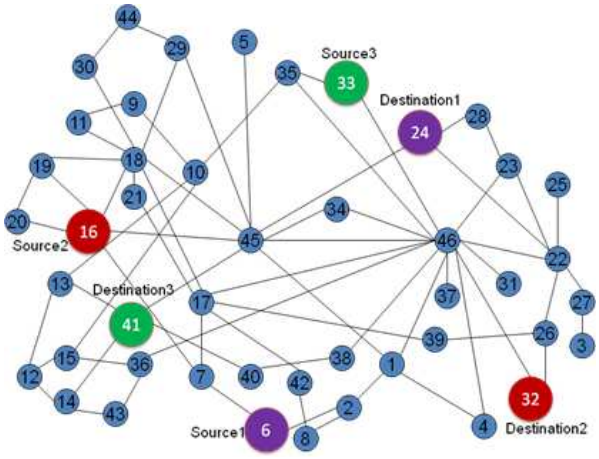

Fig. 1. The 46-node CPN testbed used in our experiments

The experiments were conducted in a real 46-node test-bed representing the SwitchLAN network topology[1]. All links have the same capacity ($10\,Mbits/s$). There are three Source-Destination (S-D) pairs that correspond to three users in the network. Each user generates UDP traffic at constant bitrate of $7Mbps$. Since the capacity of each link is $10Mbps$ this means that when failures occur the network operates at its limits and is usually highly congested. At the beginning of the experiment all nodes except the sources and destinations are vulnerable. All the sources and destinations are immune so that they will never suffer a failure.

Each experiment lasts for $120s$. The worm spread starts $10s$ after the start of each experiment and their total duration varies according to the scanning rate and the failure duration. The higher the scanning rate and the longer the failure duration, the longer the network will operate in difficult conditions and experience congestion.

[1] The Swiss Education & Research Network, http://www.switch.ch/network/

Our experiments were conducted for three routing protocols:

- the CPN protocol.
- a non adaptive version of the CPN protocol.
- the OSPF routing protocol for IP traffic.

In the first approach, CPN routes the packets of all three users with respect to minimising delay.

In the second approach by non-Adaptive CPN we mean that the CPN protocol finds the best QoS path between the source and destination nodes at the beginning of the experiment and always sends the packets through that path. In order to succeed at that, at the beginning of our experiments we run CPN as usual in order to find the paths that initially have lower delays between the sources and destinations. Just before the worm propagation starts we stop the generation of smart packets. Therefore, the network does not discover new routes and the CPN is not operating adaptively and from then on the routes stop changing.

In the third approach in order to implement IP routing in our testbed we used quangga 0.99.3 for debian linux and chose the OSPF protocol to determine the routes. The Open Shortest Path First (OSPF) is a dynamic routing protocol used in IP networks. OSPF builds routing tables based on the destination IP address found in the IP packets and it is considered to detect changes in the topology, such as link failures, and bypass them very quickly. In our experiments the OSPF costs of the links are the same for all links in the network and therefore OSPF routing converges to the minimum hop path.

We have conducted experiments for different values of scanning rates and failure durations. For example, 0.4 scanning rate means that 1 node is being scanned every 2.5 seconds, while failure duration of $60s$ means that the infected nodes will be under failure for half the time (50%) of the experiment duration and for $failure\,duration = 120s$ the nodes that become infected will stay under failure throughout the whole lifetime of the experiment. Below are the packet loss and delay results for the three users in the network throughout the duration of each experiment. Each experiment was conducted 15 times and the results presented in this paper are the average value of those runs.

Figures 2-5, show that CPN is more adaptive and performs much better than the non-adaptive CPN and the OSPF routing. In almost all cases CPN losses less data packets during the worm propagation, which means it adapts more quickly to the network changes by avoiding both the failed nodes and the congestion created by the failures. Also even if the packets in to the network are more (less packet loss), CPN manages to also keep the average delay that the users experience in smaller levels and with less fluctuations. This is due to the constant exploration of the network through the SPs which makes CPN more reliable during a worm attack.

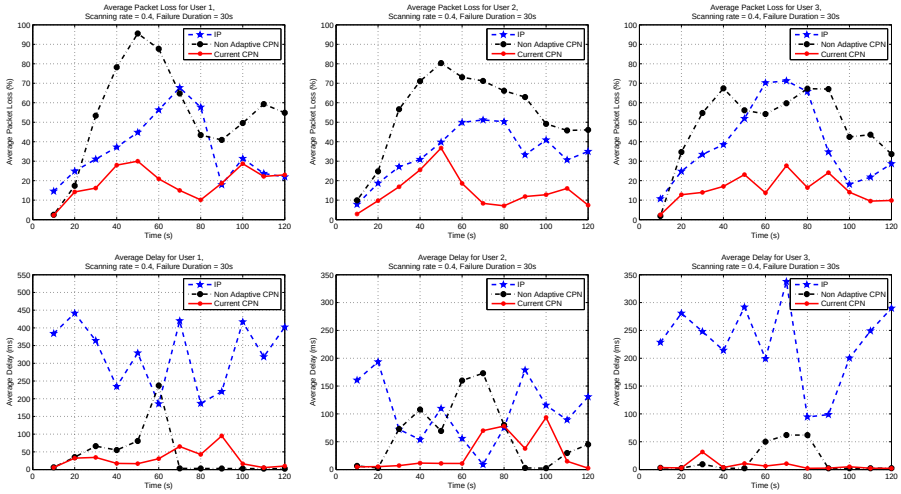

Fig. 2. Average Packet Loss and Average Delay for all 3 users when failure duration = 30s and scanning rate = 0.4 nodes/s.

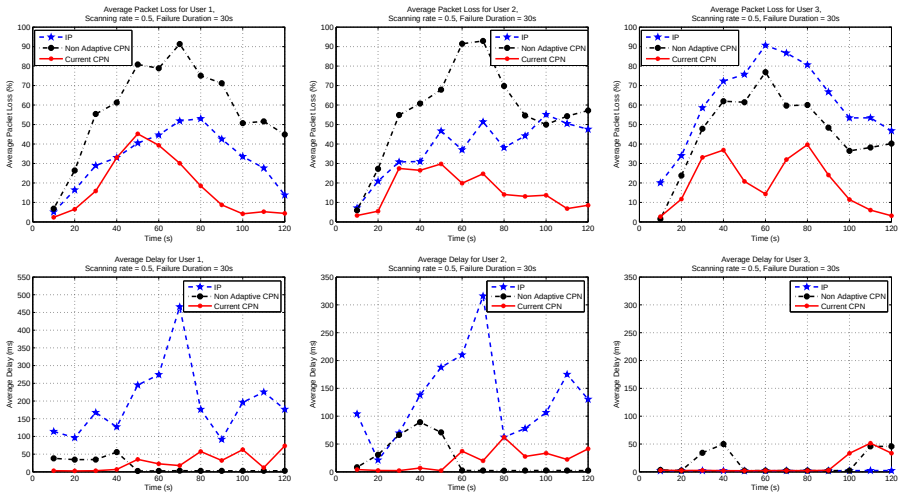

Fig. 3. Average Packet Loss and Average Delay for all 3 users when failure duration = 30s and scanning rate = 0.5 nodes/s.

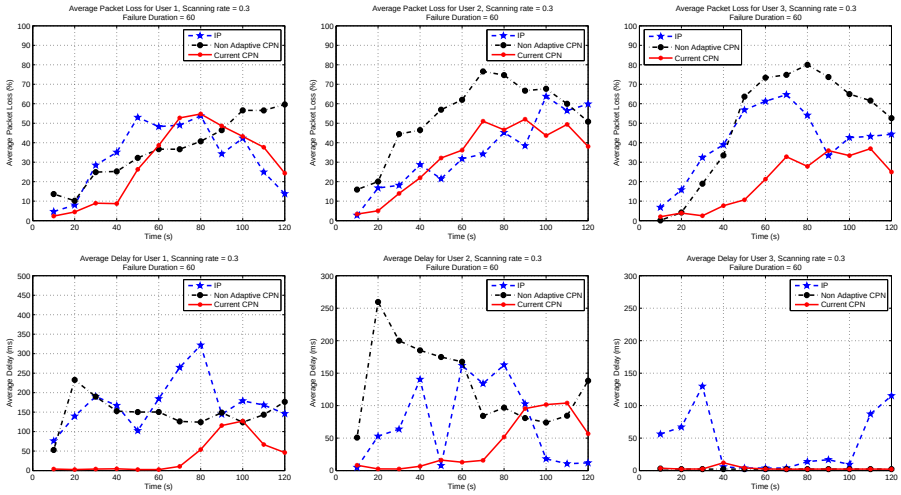

Fig. 4. Average Packet Loss and Average Delay for all 3 users when failure duration = 60s and scanning rate = 0.3 nodes/s.

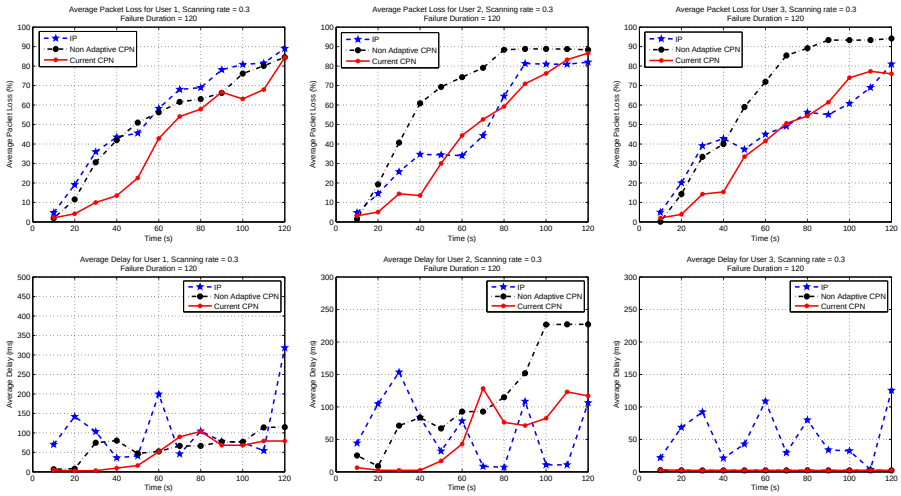

Fig. 5. Average Packet Loss and Average Delay for all 3 users when failure duration = 120s and scanning rate = 0.3 nodes/s.

But, even though CPN performs much better than the other, more conventional, routing protocols, there are some cases where it doesn't adapt quickly enough. This is due to the fact that, as described in Section 2 the weights of the RNNs in a node are updated only when an ACK packet returns to a node. Therefore if an intermediate node suffers a failure, no ACK will reach back the nodes from the destination to that node, and the weights of the neurons corresponding to the links that are affected by the failure will not be updated. In order to deal with that CPN sends a small percentage of randomly routed SPs and also each neighbour sends "hello" messages to its neighbours to check if they are still responding. But in a case where the neuron which corresponds to the node/path under failure was previously chosen a lot of times, and thus has a much higher weight than the rest of the neurons it might take a big number of random SPs to discover another path and thus, if that neuron was the most excited, the subsequent source-routed data packets will continue to follow the path under failure and will be lost until a new path is discovered. Of course since the weights of the RNN are continuously being normalised this time interval will not be extremely big but still the failure will not be avoided immediately. In order to deal with this problem and make CPN more resilient to network failures we have developed a mechanism which makes the neurons of an RNN failure-aware.

5 Failure-Aware CPN

The problem with the current detection of failures in CPN, where "hello" messages were sent to its neighbours, is that in this way at each node a neuron is excluded from a decision only if its neighbour is under failure, and doesn't take into consideration failures which could be further away and can influence the selection of a specific neuron (link). Additionally, although there is a percentage of randomly routed SPs to discover sudden changes, in some failure scenarios it might need a considerable amount of random SPs before the decision of a node changes. In order to deal with this problem we implemented a simple detection mechanism that makes the neurons of CPN more failure-aware. In our scheme a neuron (representing a possible outcome link) might be considered under failure even if the first hop neighbour node is not under failure, because it might be part of a path that has failed. Therefore, even if the failure is far away all the affected nodes will detect it.

More analytically, at each RNN and for each neuron i, the timestamp of the last SP and the last ACK that used it, are stored. If no ACK was received after sending the last SP:

$$timestamp\,of\,last\,SP\,going\,through\,i - \varepsilon < timestamp\ \ of\ \ last\ \ ACK\,coming\,through\,i$$

then the link is considered "under failure" and the neuron corresponding to this link is considered "expired" and does not participate in the calculation of the excitatory probabilities and the future decisions of the RNN. The value of ε may be different for each neuron and could depend on the average delay, under normal conditions, between the node and the destination. The neuron is just

ignored and its weights do not change so that they can be used again either after the failure restoration or if another path, which bypasses the failure, is discovered.

We have tested our mechanism in the same 46-node test-bed described in the previous Section and under the same experiment conditions and configuration. In our experiments the ε value is constant for all neurons and all nodes.

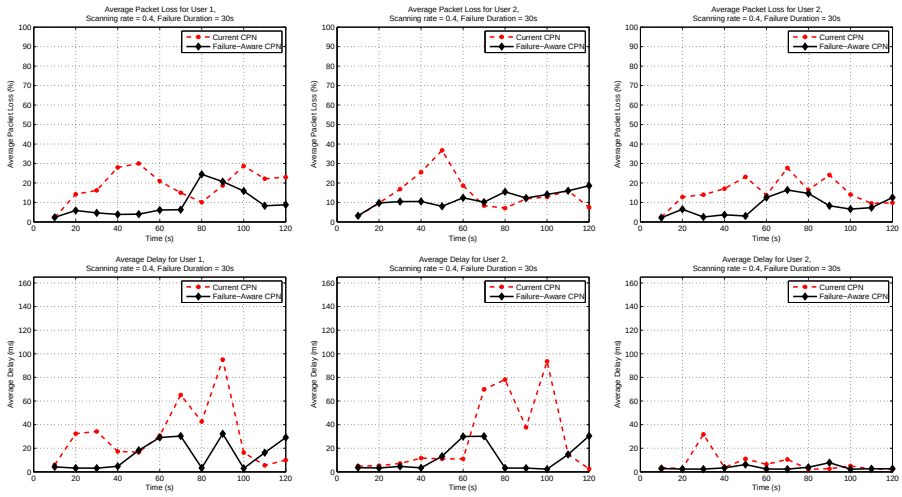

Fig. 6. Average Packet Loss and Average Delay for all 3 users when failure duration = 30s and scanning rate = 0.4 nodes/s.

As we can see from figures 6-9, the failure-aware CPN has detected the failures quicker than the current CPN. This is obvious by the fact that the average packet losses throughout the lifetime of the experiments has been reduced, meaning that failure-aware CPN was better in discovering and avoiding congestion caused by the failures. Also the ability of CPN to find the minimum delay paths was not affected since the average delay was kept in the same or lower values. We believe that the results could be further improved by finding the optimal value of the parameter ε, so that possible false detections will be avoided and the discovery of the failure is more prompt.

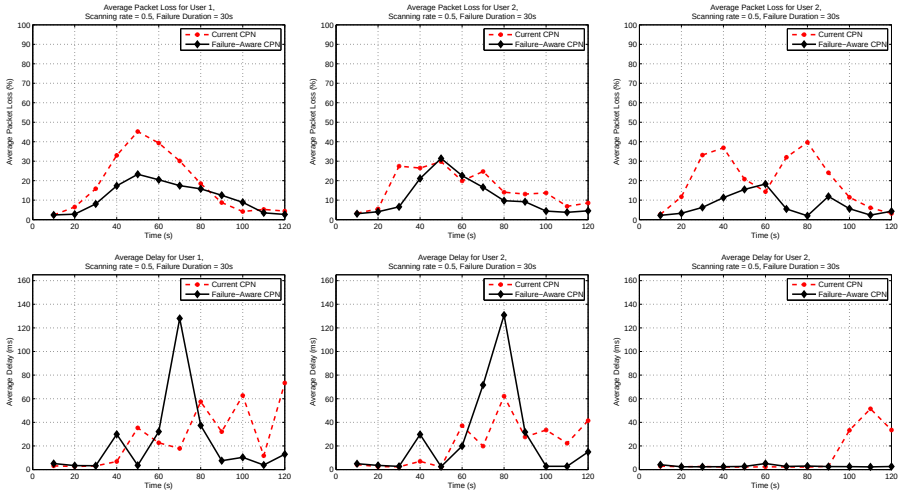

Fig. 7. Average Packet Loss and Average Delay for all 3 users when failure duration = 30s and scanning rate = 0.5 nodes/s.

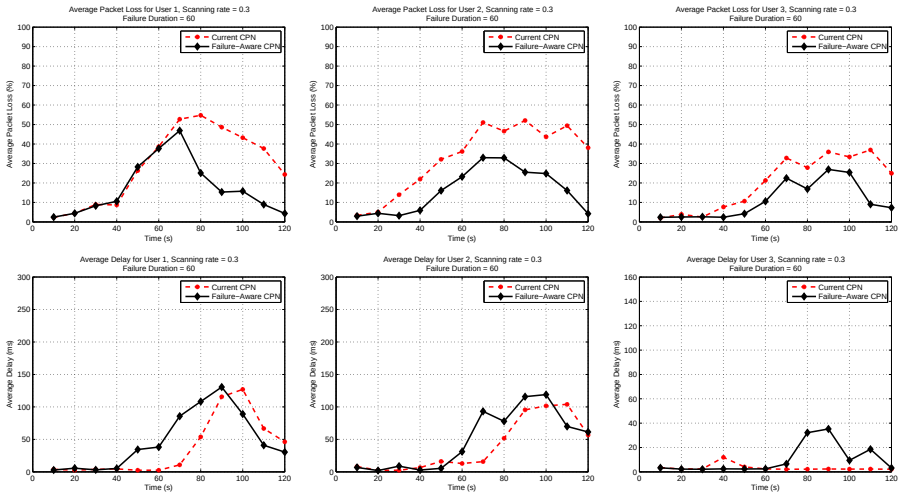

Fig. 8. Average Packet Loss and Average Delay for all 3 users when failure duration = 60s and scanning rate = 0.3 nodes/s.

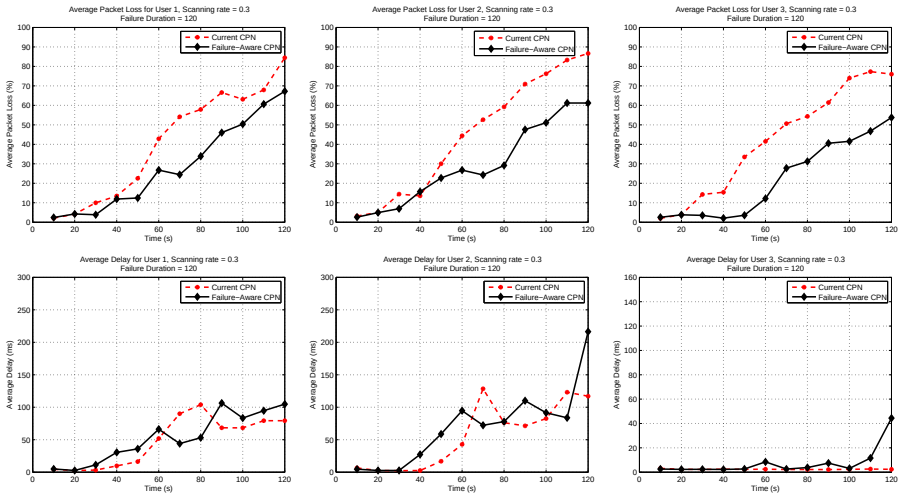

Fig. 9. Average Packet Loss and Average Delay for all 3 users when failure duration = 120s and scanning rate = 0.3 nodes/s.

6 Conclusions

We have presented experimental results that show the reliability and resilience of the CPN protocol in the presence of network worms. The experiments were conducted in a real testbed and the results demonstrate CPN's ability to guide the network during a crisis by adapting quickly to the network changes without significantly affecting the QoS provided to the users of the network. We have also described a failure detection element which is shown to further improve the performance of CPN during failures.

Further work could include experimental evaluations in scenarios of worm propagations based on epidemiological models, or mathematical models derived from empirical data of the spread of real Computer Worms. Also we intend to use our failure emulator to identify the real-time network parameters that can be used to proclaim the existence of a computer worm before it actually spreads throughout the network.

References

1. Gelenbe, E., Xu, Z., Seref, E.: Cognitive packet networks. In: Proceedings of the 11th International Conference on Tools with Artificial Intelligence (ICTAI '99), Chicago, IL, USA, IEEE Computer Society Press (1999) 47–54
2. Gelenbe, E., Lent, R., Montuori, A., Xu, Z.: Towards networks with cognitive packets. In: Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (IEEE MASCOTS), San Francisco, CA, USA (2000) 3–12 Opening Invited Paper.

3. Gelenbe, E., Lent, R., Xu, Z.: Design and performance of cognitive packet networks. Performance Evaluation **46**(2-3) (2001) 155–176
4. Gelenbe, E., Lent, R., Montuori, A., Xu, Z.: Cognitive packet networks: Qos and performance. In: Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02), Fort Worth, Texas, USA (2002) 3–9 Opening Keynote Paper.
5. Gelenbe, E., Gellman, M., Lent, R., Liu, P., Su, P.: Autonomous smart routing for network qos. In: Proceedings of the First International Conference on Autonomic Computing (ICAC), New York, NY, USA (2004) 232–239
6. Gelenbe, E.: Cognitive packet network. US Patent **6804201 B1** (2004)
7. Su, P., Gellman, M.: Using adaptive routing to achieve quality of service. Performance Evaluation **57**(2) (2004) 105–119
8. Gelenbe, E., Seref, E., Xu, Z.: Simulation with learning agents. Proceedings of the IEEE **89**(2) (2001) 148–157
9. Gelenbe, E.: Random neural networks with negative and positive signals and product form solution. Neural computation **1**(4) (1989) 502–510
10. Gelenbe, E., Liu, P., Laine, J.: Genetic algorithms for autonomic route discovery. In: Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications, Washington, DC, USA (2006) 371–376
11. Gelenbe, E., Lent, R., Xu, Z.: Measurement and performance of a cognitive packet network. Computer Networks **37**(6) (2001) 691–701
12. Gelenbe, E., Lent, R., Nunez, A.: Self-aware networks and qos. Proceedings of the IEEE **92**(9) (2004) 1478–1489
13. Gellman, M., Liu, P.: Random neural networks for the adaptive control of packet networks. In: Proceedings of the 16th International Conference on Artificial Neural Networks (ICANN 2006), Athens, Greece (2006) 313–320
14. Gellman, M.: Oscillations in self-aware networks. Proceedings of the Royal Society **464**(2096) (2008) 2169–2185
15. Gelenbe, E., Gellman, M.: Oscillations in a bio-inspired routing algorithm. In: Proceeding of the IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS 2007), BIONETWORKS Workshop, Pisa, Italy (2007) 1–7
16. Gelenbe, E., Liu, P.: Qos and routing in the cognitive packet network. In: In Proceedings of First International IEEE WoWMoM Workshop on Autonomic Communications and Computing (ACC'05), Taormina, Italy (2005) 517–521
17. Lent, R., Liu, P.: Searching for low latency routes in cpn with reduced packet overhead. In: In Proceedings of the ISCIS 2005, Advances in Computer Science and Engineering Series, Istanbul, Turkey (2005) 63–72
18. Sakellari, G., Hey, L., Gelenbe, E.: Adaptability and failure resilience of the cognitive packet network. In: DemoSession of the 27th IEEE Conference on Computer Communications(INFOCOM2008), Phoenix, Arizona, USA (2008)
19. Chen, Z., Gao, L., Kwiat, K.: Modeling the spread of active worms. In: Proceedings of the IEEE INFOCOM 2003. Volume 3., San Francisco, CA, USA (2003) 1890–1900