Adaptability and Failure Resilience of the Cognitive Packet Network

Georgia Sakellari, Laurence Hey, and Erol Gelenbe
Imperial College London
(g.sakellari, laurence.hey, e.gelenbe)@imperial.ac.uk

Abstract

Self Aware Networks (SAN) [1] is a proposal of QoS-enabled networks with enhanced monitoring and self improvement capabilities that use adaptive packet routing protocols. Such an example is Cognitive Packet Network (CPN) [2], which addresses QoS by using adaptive techniques based on online measurements. It is a distributed protocol that provides QoS-driven routing, in which users, or the network itself, declare their QoS requirements (QoS Goals) such as minimum Delay, maximum Bandwidth, minimum Cost, and so on. CPN is designed to perform self-improvement by learning from the experience of smart packets, using random neural networks (RNN) [3] with reinforcement learning (RL), and genetic algorithms.

CPN 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. As far as the decision process is concerned, each node stores a specific RNN for each QoS class, and for each active source-destination pair. Each RNN node, which represents the decision to choose a given output link for a smart packet, has as many neurons as the possible outgoing links. Decisions are taken by selecting the output link for which the corresponding neuron is the most excited. RL is carried out using a QoS Goal, such as Packet Delay, Loss, Hop Count, Jitter, and so on. 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 to data objects, and better QoS.

In the CPN protocol, SPs are generated either by a user request to create a path to some CPN node, or by a user request to discover parts of the network state, including location of certain fixed or mobile nodes, power levels at nodes, topology, paths, and their QoS metrics. To avoid overburdening the system with unsuccessful requests or packets that are in effect lost, all packets have a life-time constraint based on the number of nodes visited. In order for CPN to be more adaptive to any kind of network changes a fraction of SPs are routed at random at each intermediate router. In this way, a wider range of paths may be discovered and the network is better explored.

The performance of the CPN routing protocol has been thoroughly investigated in [4, 5, 6] and it has been shown that it is adaptable to network changes, but it has not been tested sufficiently in the presence of network failures.

Therefore, the scope of our proposed demo is to show the response of CPN during network failures. For this purpose, we have implemented a failure emulator mechanism which, according to a scenario of a failure situation, can emulate the failure of either some specific links or some machines. The failure can be specified in the scenario, it can be random, or it can follow a distribution or pattern. In our demo the failure is spread according to an epidemiological model, the AAWP (Analytical Active Worm Propagation) model. This is a discrete-time and continuous state deterministic approximation model, proposed by Chen et al. [7] to model the spread of active worms that employ random scanning.

In the AAWP model, a computer can be in one of the following states: infected, immunised, or vulnerable. At each scan the "worm" randomly chooses another host of the population and if it is immune nothing happens. If it is vulnerable it becomes infected and if it is already infected it is not re-infected (it will not change its infection behaviour). We assume that the infection delay time between two consecutive infection attempts represents the time required by a computer worm to find a server through random IP scans (regardless of whether the host is already infected or still vulnerable). So, a computer cannot infect other hosts before it is infected completely. In our implementation, the infection delay time, which is the time the worm needs to infect a machine, is currently a random value between two constant values. In the future we will try a more complex model to take into account distance between the infected node and the node it tries to infect, the degree of network congestion and other such parameters. When a node is infected it is consider under failure (traffic cannot go though it since the Ethernet interfaces are disconnected) but it can infect others. Finally, in order to capture the patching impact on the worm propagation, we dynamically immunize some hosts, by randomly choosing some non-immunized hosts (infected or simply vulnerable) to become immune. The scanning mechanism used is the random scanning mechanism but others can be used in the future, such as local subnet and topological scanning. Also, in our present application we have assumed that all hosts can reach (infect, immunise) each other directly (no topology issue).

In conclusion, our demo will show the behavior of a Self-Aware Network that uses CPN, under emulated intermittent failures generated by a failure emulator mechanism in which failure spread is modelled according to the AAWP (Analytical Active Worm Propagation) model. Through that we can see that CPN adapts quickly to failures, without significantly decreasing the QoS provided to the users of the network.

References

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

[2] E. Gelenbe, Z. Xu, and E. Seref, "Cognitive packet networks," in *Proceedings of the 11th International Conference on Tools with Artificial Intelligence (ICTAI '99)*. Chicago, IL: IEEE Computer Society Press, Nov. 1999, pp. 47–54.

[3] E. Gelenbe, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, no. 1, pp. 154–164, Jan. 1993.

[4] E. Gelenbe, R. Lent, and Z. Xu, "Measurement and performance of a cognitive packet network," *Computer Networks*, vol. 37, no. 6, pp. 691–701, Dec. 2001.

[5] E. Gelenbe, R. Lent, A. Montuori, and Z. Xu, "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, TX: IEEE Computer Society, Oct. 2002, pp. 3–12, opening Keynote Paper.

[6] E. Gelenbe, M. Gellman, R. Lent, P. Liu, and P. Su, "Autonomous smart routing for network qos," in *Proceedings of the First International Conference on Autonomic Computing (ICAC)*, New York, NY, USA, May 2004, pp. 232–239.

[7] Z. Chen, L. Gao, and K. Kwiat, "Modeling the spread of active worms," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 03)*, San Francisco, CA, USA, Apr. 2003, pp. 1890–1900.