Demonstrating Cognitive Packet Network Resilience to Worm Attacks

Georgia Sakellari
Imperial College London
Intelligent Systems & Networks Group
Dept of Electrical & Electronic Engineering
SW7 2BT, London, UK
g.sakellari@imperial.ac.uk

Erol Gelenbe
Imperial College London
Intelligent Systems & Networks Group
Dept of Electrical & Electronic Engineering
SW7 2BT, London, UK
e.gelenbe@imperial.ac.uk

ABSTRACT

The need for network stability and reliability has led to the growth of autonomic networks [2] that can provide more stable and more reliable communications via on-line measurement, learning and adaptation. A promising architecture is the Cognitive Packet Network (CPN) [5] that rapidly adapts to varying network conditions and user requirements using QoS driven reinforcement learning algorithms that drive the routing control. Contrary to conventional mechanisms, the users rather than the nodes, control the routing by specifying their desired QoS requirements (QoS Goals), such as Minimum Delay, Maximum Bandwidth, Minimum Cost, etc., and the network then routes each user's traffic individually based on their specific needs and on a "glocal" view. In CPN the user has the ability to explore the network for its own needs, and evaluate its own impact on the network as a whole and vice-versa, and then take appropriate decisions. CPN routing has been evaluated extensively under normal operating conditions and has proven to be very adaptive to network changes such as congestion. Here we show how CPN can respond and survive to catastrophic node failures caused by the spread of network worms. This survival is based on two complementary approaches that are run concurrently: one the one hand, each user attempts to concurrently and adaptively avoid paths which are infected, and secondly patching algorithms are continuously run to repair the network. Experiments show that this approach assures the stability of network communications throughout the course of an attack.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Fault tolerance, Reliability; C.2.2 [**Network Protocols**]: Routing protocols; K.6.5 [**Security and Protection**]: Invasive software (e.g., viruses, worms, Trojan horses)

General Terms

Measurement, Performance, Reliability

Keywords

Reliability, Routing protocols, Cognitive packet network, Self-aware networks, Network worms

1. THE COGNITIVE PACKET NETWORK (CPN)

CPN routing [9, 11, 10, 6, 4, 5, 21] was designed to perform self-improvement by learning from the experience of special "smart packets" (SPs) that constantly probe the network. In addition to SPs that are used for discovery, CPN also uses source routed dumb packets (DP) to carry the payload, and acknowledgement (ACK) packets to bring back information that has been discovered by SPs. This information brought back by ACKs is used in nodes to train neural networks via a Reinforcement Learning (RL) algorithm that has a relatively short memory to 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 [22]. 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. As far as the decision process is concerned, Random Neural Networks (RNNs) [3] are mainly used. The RNN is a biologically inspired 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 neurons 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 [15]. The state of a neuron, which represents the probability that the neuron is excited, satisfies 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, represents the decision to choose a given output link for a SPs. The arrival of SPs triggers the execution of RNN and the routing decision is the output link corresponding to the most excited neuron.

The learning algorithm that was designed into CPN is Reinforcement Learning (RL); this resulted from prior studies of the routing of autonomous mobile agents in a dangerous landscape [15]. RL is used to change synaptic 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.

The CPN has been shown to be effective for a variety of uses [21], including traffic balancing [13], power-based routing in mobile ad hoc networks [8] and admission control (AC) [14]. From the security aspect, the authors of [7] investigated the application of defence techniques on the resilience of the CPN against DoS attacks. They introduced a generic framework of DoS protection based on the dropping of probable illegitimate traffic, and presented a mathematical model with which one can measure the impact that both attack and defence have on the performance of a network. Their CPN-based distributed DoS defence technique exploits the ability of the CPN to trace traffic going both downstream and upstream, owing to SPs and ACK packets. When a node detects an attack, it uses the ACKs to ask all intermediate nodes upstream to drop the packets of the attack flow. Each node is allowed to select the maximum bandwidth that it will accept from any flow that terminates at the node and the maximum bandwidth that it allocates to a flow that traverses the node. These parameters may vary dynamically as a result of other conditions, and they can also be selected based on the identity and the QoS needs of the flows. When a node receives an SP or DP from a flow that it has not previously encountered (e.g. with a new source–destination pair, or a new QoS class), it sends a Flow-ACK packet back to the source along the reverse path and informs the source of its bandwidth allocation. The node monitors the flows that traverse it and drops packets of any flow that exceeds the allocation; it may also inform upstream nodes that packets of this flow should be dropped. Other possible actions include diverting the flow into a 'honeypot' or to a special network. This generic defence was further improved by using prioritization and rate-limiting instead of simple dropping [12, 19]. The same authors have also introduced a DoS detection mechanism that makes use of on-line statistics collected by the CPN protocol's monitoring system and fused them with a RNN [18]. More analytically, the scheme uses input features to capture both the instantaneous behaviour and the longer-term statistical properties of the traffic. In an off-line information gathering step, it obtains the probability density function, estimates and evaluates the likelihood ratios for the input features. During the real-time decision step it measures and calculates the features of the incoming traffic, finds the likelihood ratios corresponding to those values and aggregates these likelihood values using an RNN. The overall architecture outputs a numerical value that is a measure of having an on-going attack in the network, which is consequently used in the prioritization and rate-limiting mechanisms previously mentioned [20, 17].

2. EMULATING NETWORK WORMS

Network worms are malicious self-replicating and self- propagating applications that exploit the 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 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 can lead to serious disruption for the users of the network, due to information loss and delays.

Emulating network worms in a controlled and reproducible manner is vital for the evaluation of the resilience of a network to such scenarios. For this reason we have developed such an emulator in which the infection of a worm causes a network node to fail. This is achieved by disabling all Ethernet interfaces of a node which are connected to the network, so that no traffic can go through that node. The failure can then be restored by re-enabling these interfaces. The failure propagation can be random, according to a probability distribution or a pattern. Here, we consider failures that propagate as a computer worm that tries to infect a network's nodes. Specifically, 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. to model the spread of active worms that employ random scanning [1]. In the AAWP model, a node can be in one of the following states: infected, immunised, vulnerable. In our present application we have assumed that all hosts can reach (infect or immunise) each other directly (the topology of the network is irrelevant). 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 does not get re-infected. 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 time the worm needs to infect a machine, called the infection delay time, is a random value within a predetermined range, but the emulator could be extended so that the infection delay time could be subtracted by a more complex model which takes into account the 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 considered under failure but it can still infect others. Finally, in order to capture the patching impact on the worm propagation, we dynamically immunise some hosts, which, after some time, start immunising others (infected or simply vulnerable) randomly. The time a newly immunised node has to wait before it starts immunising others is again a random value. The scanning mechanism used is the random scanning mechanism but others could be used, such as local subnet and topological scanning.

3. OUR PROPOSED DEMO

Our demonstration will present videos of real-time experiments, conducted in a 46 node testbed located at Imperial College London, under different traffic conditions. The topology of the testbed represents the real SWITCHlan network topology (the Swiss Education & Research Network (SWITCHlan) network provides service in Switzerland to all universities, two federal institutes of technology and the major research institutes, [1]. In order to make the environment more realistic we used actual details of the 46-router backbone, complete with bandwidth, OSPF costs, and link-level delays which were given by the administrators of the SwitchLAN network to the authors of [16].

The scope of our proposed demo is to demonstrate the response and stability of CPN under emulated worm spreads and intermittent failures generated by a failure emulator mechanism, in which failure spread is modelled according to the AAWP (Analytical Active Worm Propagation) model. We will show that CPN adapts quickly to failures, without significantly decreasing the measured QoS provided to the users of the network.

4. REFERENCES

[1] Z. Chen, L. Gao, and K. Kwiat. Modeling the Spread of Active Worms. In *Proceedings of the IEEE INFOCOM 2003*, volume 3, pages 1890–1900, San Francisco, CA, USA, Apr. 2003.

[2] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli. A survey of autonomic communications. *ACM Trans. Adapt. Autonomous Systems (TAAS)*, 1(2):223–259, 2006.

[3] E. Gelenbe. Random Neural Networks with Negative and Positive Signals and Product Form Solution. *Neural computation*, 1(4):502–510, 1989.

[4] E. Gelenbe. Cognitive Packet Network. *US Patent*, 6804201 B1, Oct. 2004.

[5] E. Gelenbe. Steps towards self-aware networks. *Communications of the ACM*, 52(7):66–75, July 2009.

[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)*, pages 232–239, New York, NY, USA, May 2004.

[7] E. Gelenbe, M. Gellman, and G. Loukas. An Autonomic Approach to Denial of Service Defence. In *Proceedings of First International IEEE WoWMoM Workshop on Autonomic Communications and Computing (ACC'05)*, pages 537–541, Taormina, Italy, June 2005.

[8] E. Gelenbe and R. Lent. Power-aware ad hoc cognitive packet networks. *Ad Hoc Networks Journal*, 2(3):205–216, July 2004.

[9] E. Gelenbe, R. Lent, A. Montuori, and Z. Xu. Towards Networks with Cognitive Packets. In *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (IEEE MASCOTS)*, pages 3–12, San Francisco, CA, USA, Aug. 2000. Opening Invited Paper.

[10] 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)*, pages 3–9, Fort Worth, Texas, USA, Oct. 2002. Opening Keynote Paper.

[11] E. Gelenbe, R. Lent, and Z. Xu. Design and Performance of Cognitive Packet Networks. *Performance Evaluation*, 46(2-3):155–176, Oct. 2001.

[12] E. Gelenbe and G. Loukas. A Self-Aware Approach to Denial of Service Defence. *Computer Networks*, 51(5):1299–1314, Apr. 2007.

[13] E. Gelenbe and A. Nunez. Traffic Engineering with Cognitive Packet Networks. *Simulation Series*, 35(4):514–518, Apr. 2003.

[14] E. Gelenbe, G. Sakellari, and M. D' Arienzo. Admission of QoS Aware Users in a Smart Network. *ACM Transactions on Autonomous and Adaptive Systems*, 3(1):4:1–4:28, Mar. 2008.

[15] E. Gelenbe, E. Seref, and Z. Xu. Simulation with Learning Agents. *Proceedings of the IEEE*, 89(2):148–157, Feb. 2001.

[16] M. Gellman and P. Liu. Random Neural Networks for the Adaptive Control of Packet Networks. In *Proceedings of the 16th International Conference on Artificial Neural Networks (ICANN 2006)*, pages 313–320, Athens, Greece, Sep. 2006.

[17] G. Loukas and G. Oke. Likelihood Ratios and Recurrent Random Neural Networks in Detection of Denial of Service Attacks. In *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2007)*, pages 16–18, San Diego, California, USA, July 2007.

[18] G. Oke and G. Loukas. A Denial of Service Detector based on Maximum Likelihood Detection and the Random Neural Network. *The Computer Journal*, 50(6):717–727, Sep. 2007.

[19] G. Oke and G. Loukas. Distributed Defence Against Denial of Service Attacks: A Practical View. In *Proceedings of 1st BCS International Academic Conference, Visions of Computer Science*, pages 153–162, London, UK, Sep. 2008.

[20] G. Oke, G. Loukas, and E. Gelenbe. Detecting Denial of Service Attacks with Bayesian Classifiers and the Random Neural Network. In *Proceedings of the IEEE International Fuzzy Systems Conference (FUZZ-IEEE 2007)*, pages 1964–1969, London,UK, July 2007.

[21] G. Sakellari. The Cognitive Packet Network: A Survey. *The Computer Journal: Special Issue on Random Neural Networks, doi:10.1093/comjnl/bxp053*, June 2009.

[22] P. Su and M. Gellman. Using adaptive routing to achieve Quality of Service. *Performance Evaluation*, 57(2):105–119, June 2004.

[1]http://www.switch.ch/network/