Cognitive Packets in Large Virtual Networks

Ricardo Lent and Erol Gelenbe

Abstract Network testbeds are useful for protocol performance evaluation. They overcome most challenges commonly involved in live network experimentation, retaining fair realism and repeatability under controllable conditions. However, large-scale testbeds are difficult to set up due to limited resources available to most experimenters. In this paper, we explore the use of hardware virtualisation as an experimental tool to improve resource efficiency, allowing to boost the effective number of nodes available for network testing. By virtualising network routers and links, a cluster environment with off-the-shelf equipment can host hundreds of virtual routers for large-scale network testing. We apply this technique to construct a 800-node Cognitive Packet Networks (CPN) testbed that provides insight into the benefits and limitations of the approach.

1 Introduction

Testbed experimentation with network prototypes can complement, and sometimes replace modelling and simulation studies [9], bringing a closer-to-reality alternative to evaluation studies in network research. Repeatable experiments can stress working conditions for protocol implementations to verify operation aspects and performance limitations. Cognitive packet network (CPN) routing [3] has been evaluated in several testbeds [4, 5, 7], to verify the effectiveness of the approach. These tests have provided useful feedback to improve the prototype and algorithm, and

R. Lent (✉) · E. Gelenbe
Department of Electrical and Electronic Engineering,
Intelligent Systems and Networks, Imperial College,
London SW7 2AZ, UK
e-mail: r.lent@imperial.ac.uk

E. Gelenbe
e-mail: e.gelenbe@imperial.ac.uk

E. Gelenbe and R. Lent (eds.), *Computer and Information Sciences III*,
DOI: 10.1007/978-1-4471-4594-3_2,

inspiration for new research. However, tests have been limited so far to relatively small networks, of tens of nodes (less than 100 in all cases) connected with a proportional number of links. Tests on larger networks are always desirable, not only because they permit observations of the network operation under conditions closer to production systems, but also useful for debugging and measuring practical limits of given implementations. Despite its drawbacks for network performance testing (e.g., unrealistic packet delays), virtualisation can be a viable alternative for creating a high number of virtual devices, which could be used for large-scale network testing under certain conditions. It is interesting to note that forms of virtualisation have been supported by networks for a while through diverse techniques, such as Virtual LANs (VLANs) and tunneling (e.g, L2TP, PPTP). Virtual machine managers on the other hand, such as VMware, VirtualBox and QEMU/KVM, support network virtualisation for guest machines providing emulated network domains. Some recent proposals have suggested conceptual extensions to virtualisation to improve the control and management of physical networks [2, 11]. Unfortunately, these techniques are of limited use to handle in practice the thousands of links and nodes that a large-scale testbed might require. For example, in the case of VLANs, the VLAN ID (12 bits) in IEEE 802.1p limits the identification of 4,096 VLANs. In practice, most commercial switches support about 250 VLANs. Similarly, current virtual machine hypervisors usually limit the number of virtual networks to 8. While likely enough to satisfy the networking needs of most sites, these numbers are clearly too modest for our purposes. On the other hand, several initiatives, such as FIRE (http://www.ict-fire.eu) and GENI [1] seek to build remotely accessible large experimental facilities. A few notable examples of existing facilities are Emulab [15] and PlanetLab. Having remote access to testbeds is an advantage but their shared nature creates restrictions to the kind and size of experiments that could be carried out.

Recent work has addressed optimisation issues resulting from embedding virtual topologies in physical ones [17]. For a single site, a simpler approach is to leverage platform virtualisation to create router instances in a similar way machine instances can be created with an hypervisor. Unlike the approach in [10], our plan is to enforce logical links without tunnels which provides greater scalability. Naturally, because of the virtual nature of the network, packet traveling times may not accurately follow those on a physical network [16]. Therefore, a careful selection of network metrics will be needed to produce meaningful results from experiments under virtualisation. For clarity, we will elaborate our approach to network testing with virtualisation within the specific context of Cognitive Packet Networks and apply it to evaluate the network under router misbehaviour. The approach is however easily applicable to a wider range of network tests.

2 Workflows in a Virtual Network

Setting up a new virtual network involves two well-defined parts: creating nodes and enforcing the virtual topology (i.e., creating edges). For the experiment, we make use of machine virtualisation to instantiate network nodes given that the existing CPN

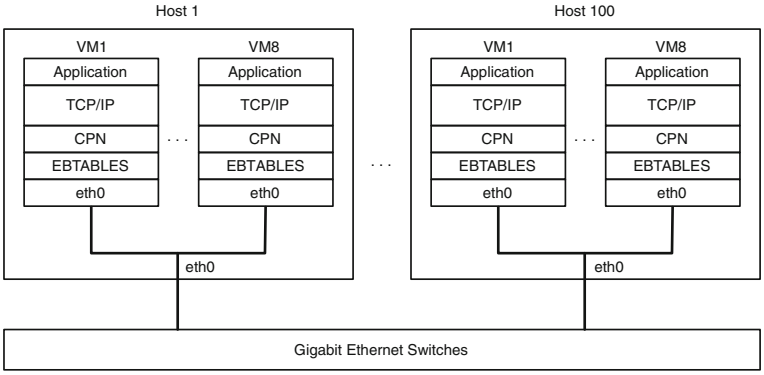

Fig. 1 Virtual network testbed for CPN

prototype works as a Linux kernel module. The module requires a virtualisation platform able to create virtual machine (VM) instances that are capable of running a Linux kernel. Because of the reduced size of the prototype—about 64 KB, Linux and the CPN module can easily run in virtual machines with RAM disks. Using RAM disks can greatly simplify the deployment process because no disk images would need to be created across the physical testbed. Via the Preboot Execution Environment (PXE), machines instances can quickly obtain both a kernel with the CPN extension and a RAM disk image from a server machine. Alternatively, at node creation time, full disk images must be created (typically, replicated from a master copy). In practice, we developed UNIX scripts to automate virtual machine creation with full pre-copied images. The script allowed a rapid creation of virtual routers, at about 100 machines per minute.

Assuming the use of a single physical network for virtual machine interconnection with bridged networking, edge virtualisation can be implemented through packet filtering, similarly to [12]. The only constraint is that virtual machines would need to be created with a consistent and predictable MAC address, which could be enforced in the VM creation process. An arbitrary network topology can then be implemented through bride firewalling, by setting appropriate filtering rules at each node to replicate the desired topology. Our approach relies on `EBTABLES`, which can enforce firewall rules at the MAC layer. The resulting process becomes transparent to all networking layers above the MAC layer, including the CPN layer (see Fig. 1). We were able to test this approach with hundreds of firewall rules without a noticeable drop in performance.

2.1 A 800-Router CPN Testbed

The topology was modeled after the Internet autonomous-system (AS) graph collected by the Internet maps [18] project. We selected a map of 2009 because of its

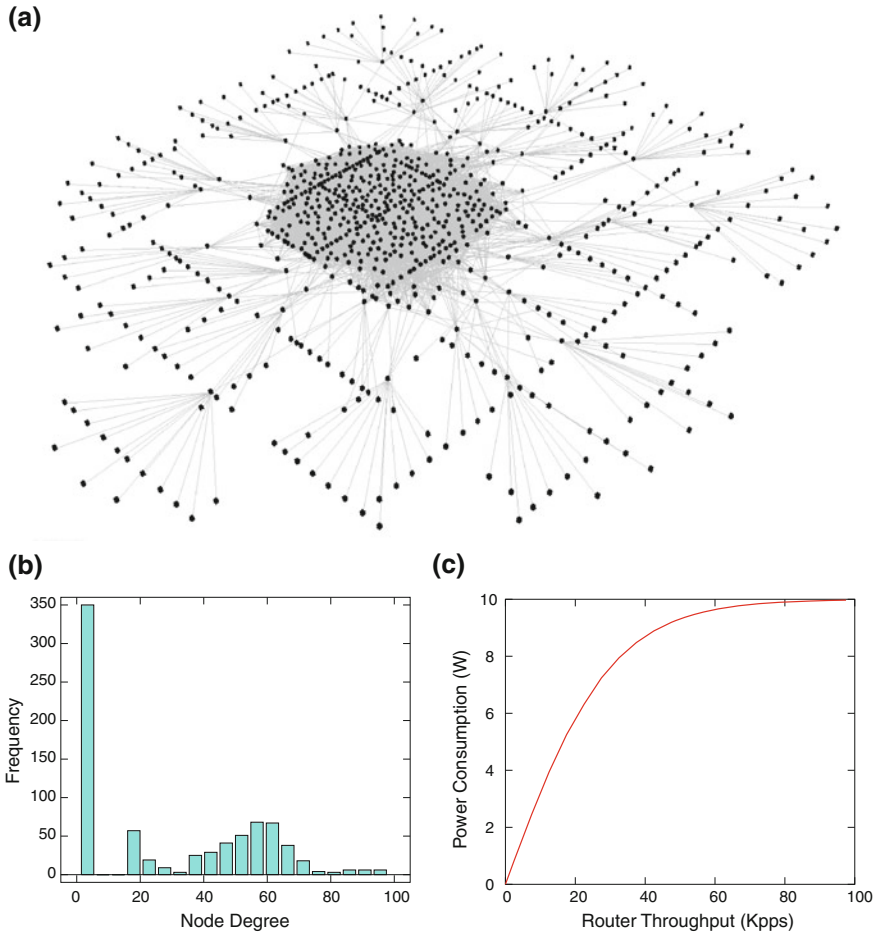

Fig. 2 A 800-node CPN testbed. **a** Topology. **b** Node degree distribution. **c** Power profile of routers

smaller size compared to current maps. To define the topology, we included all Tier 1 and large Internet Service Provider nodes from the map collection, each represented by a CPN router. Nodes with very large number of neighbors were transformed into mesh subnets for practical reasons (i.e., because of the maximum number of neighbours that a CPN node could handle). In addition, 350 end nodes were connected to randomly selected routers. The whole network consisted of 800 nodes and 11,762 links forming the topology shown in Fig. 2a. Node degree (i.e., number of neighbours) frequency is plotted in Fig. 2b. Clearly, end hosts (350 of them) have degree 1 and routers have varying number of neighbours, up to 100, which was the limit assumed in the large node division. The physical system consisted of 100 identical machines (Fig. 1): 8-core Xeon X3450 2.67 GHz and 8 GB of RAM with Gigabit

Ethernet connections through Cisco Nexus 2148T switches forming a star topology with 10 Gbps connections to a central switch—a Cisco Nexus 5010. Each physical machine hosted exactly 8 virtual machines (VM) with a VirtualBox hypervisor. Each VM was configured as a single-core machine with 256 MB of RAM. Identical copies of a disk image containing an extended Ubuntu Linux kernel with a CPN implementation were attached to each virtual machine. The disk image also contained a topology file and scripts to enforce a MAC-layer filtering based on the MAC address of each node and expected neighbours listed in the topology file. Therefore, each node configuration was customized based on its MAC address assigned at VM creation time. MAC addresses followed a certain convention which allowed node to use them in determining their IP address (for control) and CPN address (for testing). In addition, each CPN router was associated with a (simulated) power consumption dependent on the packet total throughput handled by the node as depicted in Fig. 2c. Such response can be measured by profiling the system with a power measuring tool [13].

2.2 Studying Router Misbehavior

To evaluate the testbed, we applied it to measure CPN performance under router misbehaviour, which could result from nodes that have been compromised by an intruder. Such nodes would behave in an undesirable manner, affecting user flows. We are interested in quantifying how such flows could be affected. To conduct the evaluation, we considered either 50, 100 or 150 concurrent flows for any given experiment run, randomly selecting nodes from the set of 350 end nodes that we defined when creating the network topology. Similarly, we select a certain number of routers to misbehave during the experiment to be either 5, 23, 45, 90 or 180 for any particular run. In addition, we define a target rate for each of the flows to be 10, 50, 100, 150, 200, 250 or 300 pps with 1 KB-long packets. All flows are UDP without retransmission at upper layers. Two types of router misbehaviour were examined. In the first case, misbehaving routers drop all incoming traffic (including smart packets), so that those routers could potentially break network connectivity. This is the simplest case of network intrusion where the attacker has managed to disable the router. In a second case, we consider that the misbehaving router will send user packets to the worst possible next hop as a possible consequence of a more elaborate attack. Because multiple virtual machines share the same physical resources (e.g., the same network port), latency measurements could not reflect the actual values on a corresponding physical network. We have selected power consumption in nodes [6–8] as a metric for the routing goal of smart packets. Power consumption in real routers has a significant idle component and a dynamic component, being the latter quite relevant for software routers [14]. For testing purposes, we zeroed the idle component (see Fig. 2c) to direct the routing optimisation towards the lowest dynamic consumption due to user flows. Measurements for the first type of router misbehaviour are depicted in Fig. 3 comparing the level of average power consumption observed in the network under

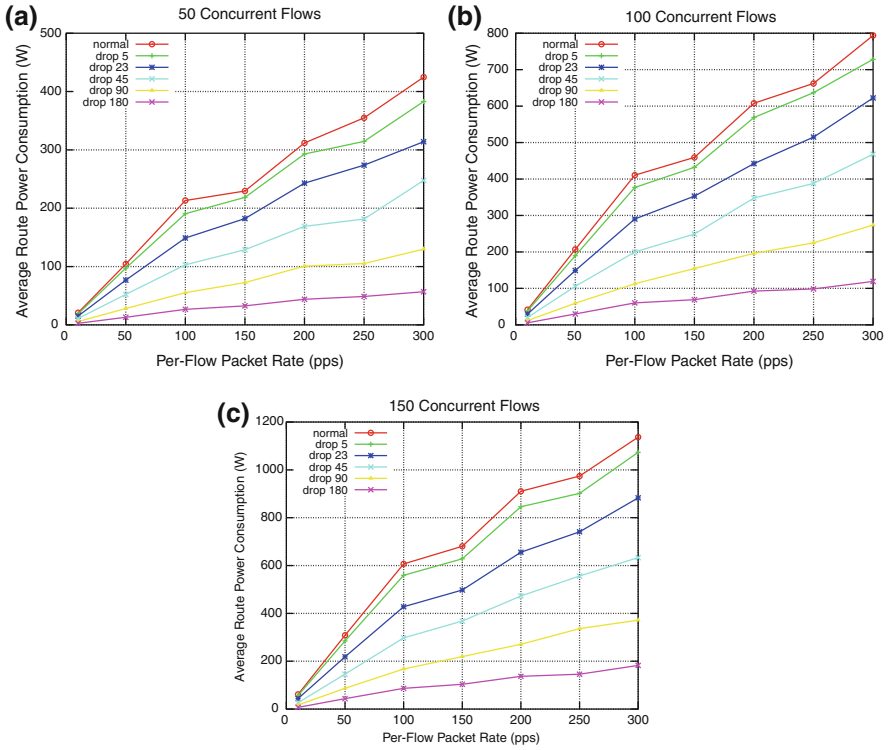

Fig. 3 Average power consumption of paths

different number of misbehaving routers. The normal behaviour is also included (top curve) for comparison purposes. Randomly choosing routers to drop packets may cause network partition, which explains the high packet loss that can be observed in cases of drop misbehaviour.

A higher number of packet-drop nodes tends to diminish the overall network power consumption because of the higher level of packet loss as can be inferred from Fig. 4b. This is because no packet retransmission was considered. On the other hand, we observed path lengths to be approximately similar across different combinations of number of flows. However, there was a significant difference in the path length of packets between the two types of router misbehaviour, likely expected given the differences in the effective network topology.

It is important to note that the purpose of the tests was mainly the validation of the CPN prototype in a large network setting rather than providing a specific solution to a network intrusion problem. Nevertheless, we observed that CPN could easily handle to some success (all but network partitions), these types of router misbehavior without any change in the algorithm. In the case of packet drops, smart packets could avoid misbehaving nodes because they are quickly removed from the path discovering process whenever they move to a compromised router. Similarly, smart packets tried

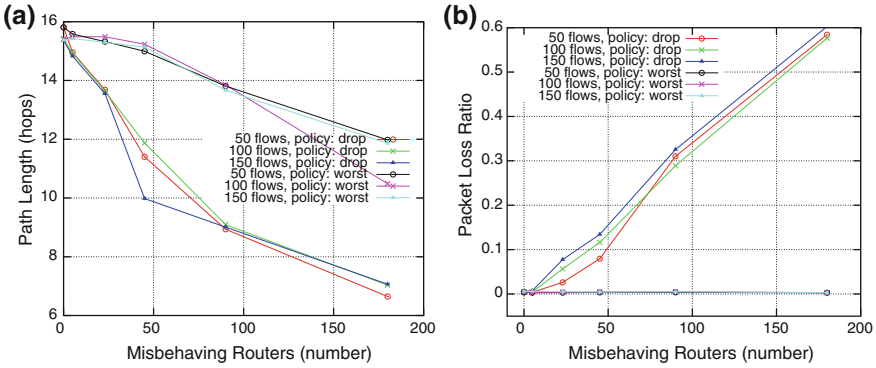

Fig. 4 Average path length and packet loss ratio. **a**, **b** System A

to avoid routers which forward packets to the worst next hop because of their low rating reflected in mailboxes.

3 Conclusions

We have successfully tested CPN in a large network of 800 routers which suggests a viable direction to create even larger testbeds in the future. Two practical considerations resulted from our experiments. First, the CPN implementation was able to handle up to 250 concurrent neighbours per node. This limit is mainly due to the use of static memory allocation in the CPN kernel implementation. While the neighbour table implementation could be modified to make use of dynamic memory allocation and increase this limit, we observed that the main bottleneck was rather the high memory usage of the random neural network (see [5]) which requires two matrices of size the square number of neighbours. A possible solution is to dynamically assign neighbours to the matrix, for example, in a least-recently used fashion, so as to limit its size by only considering the most used (or successful) neighbours in the table. A future work will address these implementation issues in further detail.

Acknowledgments This work has been possible in part thanks to the support provided by the UK Technology Strategy Board/EPSRC SATURN Project, the FP7 FIT4Green Project and the computing facilities at Northrup-Grumman UK Ltd.

References

1. Anderson, T., Reiter, M.K.: Geni: global environment for network innovations distributed services working group (2006). http://www.homes.cs.washington.edu/~tom/support/GDD-06-24.pdf
2. Casado, M., Koponen, T., Ramanathan, R., Shenker, S.: Virtualizing the network forwarding plane. In: Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow, PRESTO '10, pp. 8:1–8:6. ACM, New York (2010)

3. Gelenbe, E.: Cognitive packet network. US Patent 6,804,201, 2004
4. Gelenbe, E.: Steps towards self-aware networks. Commun. ACM **52**(7), 66–75 (2009)
5. Gelenbe, E., Lent, R., Xu, Z.: Design and performance of cognitive packet networks. Perform. Eval. **46**(2, 3), 155–176 (2001)
6. Gelenbe, E., Mahmoodi, T.: Energy-aware routing in the cognitive packet network. In: International Conference on Smart Grids, Green Communications, and IT Energy-Aware Technologies (Energy 2011), Venice, Italy 2011
7. Gelenbe, E., Morfopoulou, C.: A framewok for energy aware routing in packet networks. Comput. J. **54**(6), 850–859 (2011)
8. Gelenbe, E., Morfopoulou, C.: Gradient optimisation for network power consumption. In: GreenNets 2011, The First International Conference on Green Communications and Networking, Colmar, France 2011
9. Gelenbe, E., Stafylopatis, A.: Global behavior of homogeneous random neural systems. Appl. Math. Model. **15**(10), 534–541 (1991). doi:10.1016/0307-904X(91)90055-T
10. Jiang, X., Xu, D.: Vbet: a vm-based emulation testbed. In: Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research, MoMeTools '03, pp. 95–104. ACM, New York (2003)
11. Keller, E., Rexford, J.: The "platform as a service" model for networking. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, INM/WREN'10, p. 4. USENIX Association, Berkeley (2010)
12. Lent, R.: Design of a MANET testbed management system. Comput. J. **49**(4), 171–179 (2006)
13. Lent, R.: A sensor network to profile the electrical power consumption of computer networks. In: Proceedings of GLOBECOM Workshops (GC Wkshps), pp. 1433–1437, Miami, 2010. doi:10.1109/GLOCOMW.2010.5700175
14. Lent, R.: Simulating the power consumption of computer networks. In: Proceedings of the 15th IEEE International Workshop on Computer Aided Modeling Analysis and Design of Communication Links and Networks (IEEE CAMAD), Miami, 2010
15. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. In: Proceedings of the Fifth Symposium on Operating Systems Design and Implementation, pp. 255–270. USENIX Association, Boston (2002)
16. Whiteaker, J., Schneider, F., Teixeira, R.: Explaining packet delays under virtualization. SIGCOMM Comput. Commun. Rev. **41**(1), 38–44 (2011)
17. Xin, Y., Baldine, I., Mandal, A., Heermann, C., Chase, J., Yumerefendi, A.: Embedding virtual topologies in networked clouds. In: Proceedings of the 6th International Conference on Future Internet Technologies, CFI '11, pp. 26–29. ACM, New York (2011)
18. Zhang, B., Liu, R., Massey, D., Zhang, L.: Collecting the internet as-level topology. SIGCOMM Comput. Commun. Rev. **35**, 53–61 (2005). doi:10.1145/1052812.1052825. http://www.doi.acm.org/10.1145/1052812.1052825

http://www.springer.com/978-1-4471-4593-6

Computer and Information Sciences III
27th International Symposium on Computer and Information Sciences
Gelenbe, E.; Lent, R. (Eds.)
2013, XII, 512 p., Hardcover
ISBN: 978-1-4471-4593-6