

# Topology Generation, Instrumentation, and Experimental Control Tools for Emulation Testbeds

Roman Chertov, Sonia Fahmy, Pankaj Kumar, David Bettis, Abdallah Khreishah, Ness B. Shroff  
Purdue University

## I. INTRODUCTION

Any network experiment requires three key components: (i) topology generation, (ii) control, and (iii) instrumentation and data collection. Network topology generation and route configuration is the most difficult component of the three. The generation of realistic yet as-small-as-possible experimental topologies remains an open research problem. Control and data collection are equally important but are more straightforward, and need to be tailored for the specific experimental environment. In simulators, control and data collection are trivial, but this is not the case on real test networks. Our tools were built for testbeds such as DETER, Emulab, or WAIL, containing physical PCs running production operating systems.

The remainder of this paper is organized as follows. Section II describes the topology generation and router configuration tools we have developed. Section III explains our event control system. Section IV describes our data acquisition tools. Finally, Section V concludes the paper.

## II. TOPOLOGY GENERATION AND ROUTER CONFIGURATION

It is imperative to have representative benchmarks including Internet topology data (which is constantly evolving) continuously available for the security research community. Towards this end, we are developing a tool suite that makes it easy to use real or generated topologies with dynamic (intra-domain and inter-domain) routing on DETER.

The first tool in our suite is similar to RocketFuel [9] from the University of Washington. RocketFuel includes components for alias resolution, and for inference of several routing (e.g., Open Shortest Path First (OSPF) routing weights) and geographical (e.g., location) properties. A few of the components of RocketFuel as well as several sample topologies are available through the ScriptRoute [10] project and the RocketFuel web pages, but the complete tool is not available for download. Our tool, which we refer to as *NetTopology*, invokes a limited number of traceroute commands from different traceroute servers [2] and synthesizes the routes and latency information.

Configuring routers running the Border Gateway Protocol (BGP) poses a significant challenge, since Internet Service Providers (ISPs) use complex BGP policies for traffic engineer-

ing. We utilize the work by Gao et al. [6], [13] to infer Autonomous System (AS) relationships, and use that information to configure BGP routers. To obtain the AS relationships, we use information made available by the University of Oregon RouteViews project. We utilize the RouteViews Cisco Format tables, and files from the straightenRV tool that is discussed on the RouteViews web page. Two of the files output by straightenRV are used (.full and .as). Our tool outputs a map from each pair of ASes to the relationships they have.

In order to generate benchmarks that can be directly used on a testbed like DETER, we have developed two additional tool sets: (i) *RocketFuel-to-ns* which converts topologies generated by RocketFuel-like tools to DETER-compliant scripts, and (ii) *RouterConfig* a router configuration script suite that can be used to configure routers (e.g., PCs running routing software) to run BGP and OSPF with the appropriate parameters according to their roles in the topologies.

*RocketFuel-to-ns* allows the user to specify a set of Autonomous Systems on the command line, or we perform breadth-first traversal of the topology graph from a specified AS number, with specified degree bounds, and a specified number of nodes bound. This enables the user to select topologies of only tens of nodes up to a few hundred nodes out of very large topologies. Figure 1 depicts an example topology generated by *RocketFuel-to-ns*, captured from the DETER testbed interface.

The *RouterConfig* tool suite can be used to configure routers both in (a) topologies based on real Internet data, and in (b) topologies generated from the GT-ITM topology generator [14]. We have selected GT-ITM since it generates representative topologies, even when the number of nodes in the topology is small [11]. In fact, a key problem we are investigating is the scale-down of a topology of several thousand or even millions of nodes to a few hundred nodes (which is the number of nodes typically available on a testbed like DETER).

In the case of a GT-ITM topology, *RouterConfig* classifies the nodes of the GT-ITM topology as OSPF routers, BGP routers, or non-router nodes. It also specifies the domain the node belongs to and the type of that domain (transit or stub).

The router configuration files that *RouterConfig* generates can be executed when the experimental node boots or reboots. The average time required to edit the configuration files manually for an experiment with 40 nodes is about 2–3 hours, because the process requires setting IP addresses for every node in the files. Using our *RouterConfig* tool, it only takes a few seconds for the process to complete.

Figure 2 gives a data flow diagram that illustrates the inputs and outputs of our topology generation and router configuration tools for emulation testbeds like DETER.

– This research has been sponsored in part by NSF/DHS grant 0335247, NSF grant 0523249, AFRL, and USC-ISI.

– Roman Chertov, Sonia Fahmy, Pankaj Kumar, and David Bettis are with the Department of Computer Science, 250 N. University St., West Lafayette, IN 47907–2066, USA. Tel: +1-765-494-6183. Fax: +1-765-494-0739, E-mail: {rchertov,fahmy}@purdue.edu. Ness B. Shroff and Abdallah Khreishah are with the School of Electrical and Computer Engineering, 465 Northwestern Ave., West Lafayette, IN 47907–2035, USA.

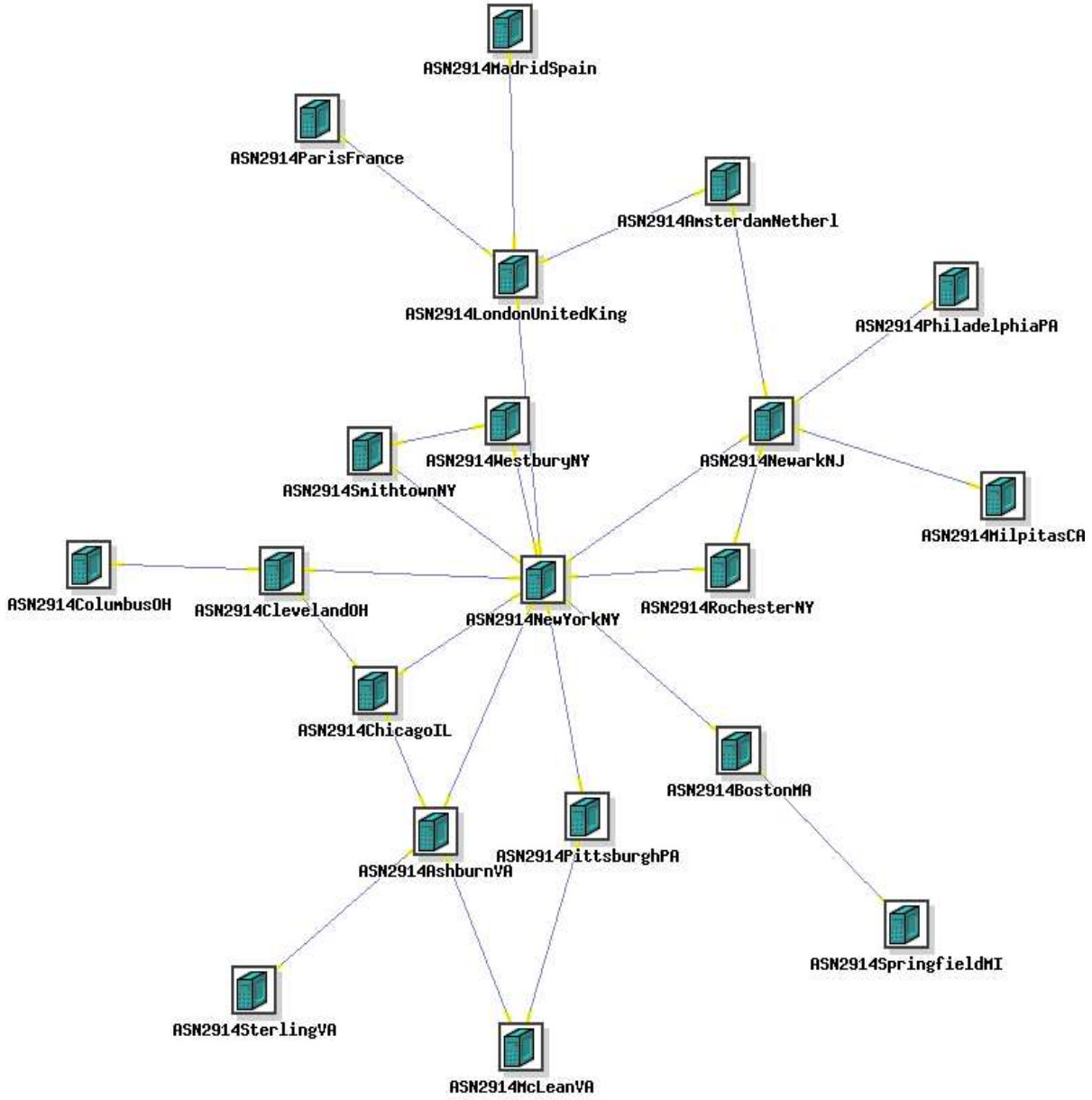


Fig. 1. A sample topology on the DETER testbed.

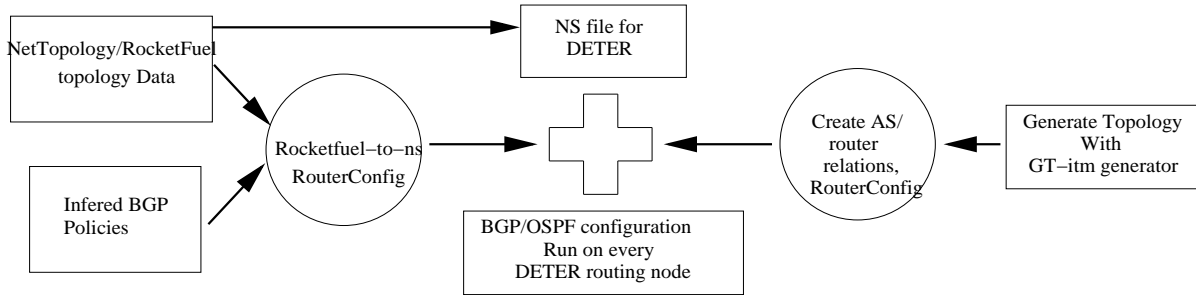


Fig. 2. Topology generation and router configuration tools data flow diagram.

### III. EVENT CONTROL SYSTEM

In network simulators such as ns-2 [12], GTNetS [1], iSSF/iSSFNet[8], and OPNET [3], it is easy to create a topology, assign tasks to the nodes, and monitor every single packet. A basic testbed – without any software support that mirrors some of these capabilities – is limited in its usefulness, since it requires the experimenters to be experts in system-level programming. Achieving the same level of control provided by a simulator on physical testbed machines is a significant undertaking. Basic topology creation capabilities are provided by emulation testbeds, such as Emulab and DETER, but an experimenter only acquires bare machines that form the desired topology, without any tools running on them.

A natural approach to describe the tasks that must be performed on the testbed nodes is to use event scripts, much like events in an event-driven simulator. The Emulab software implements a few event types such as link failures; however, most of the interaction with the nodes must be performed via a secure shell (SSH) session. We have designed a flexible mechanism to control all test machines from a central location, since manually using each computer is impossible, especially when timed events are involved. We have developed a multi-threaded utility, which we refer to as a *Scriptable Event System*, to parse the script of timed events and execute it on the test machines (communicating with them on the *control network*). Our utility is capable of receiving *callbacks* for event synchronization.<sup>1</sup> Figure 3 depicts the architecture of our system.

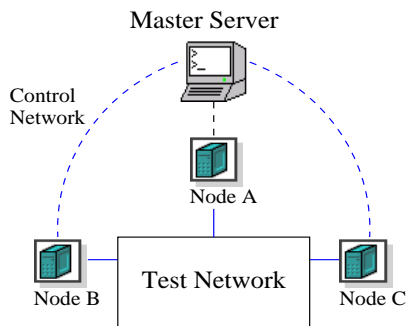


Fig. 3. Master/Zombie control network.

### IV. MEASUREMENT TOOLS

This section describes the measurement tools we have created and the packet generation utility which we have used to verify our measurements.

#### A. Host Statistics

Instrumentation and measurement on a testbed pose a significant challenge. The capability to log and correlate different types of activities and events in the test network is essential. Not only are packet traces important, but also system statistics must be measured during very high loads. We have developed a set of tools to log events on the test nodes on a per second basis. Statistics such as CPU utilization, packets per second,

<sup>1</sup>This software can be freely downloaded from <http://www.cs.purdue.edu/~fahmy/software/emist/>

and memory utilization are logged to the local disk for later manipulation. We have the capability to collect measurements on Linux nodes via system files and also query Cisco routers using SNMP from a PERL script. Scripts for measuring, merging, and plotting system data are also available for download.

#### B. Link Monitoring

In simulators, it is straightforward to monitor a link and collect statistics such as packet rates and traffic. In an emulation environment such as DETER or Emulab, we need to create our own tool to do so. The link monitor is constructed out of two components (Figure 4). The first component mirrors all passing traffic to a logging node. The second component is the logger which executes *tcpdump*. Separation of packet duplication and logging ensures that there is no competition between the two tasks. We have created the first component (the mirror) by modifying the Linux bridge module and also by using the Click modular router [7]. The mirror/logger can be added between the two experimental nodes transparently so that the nodes are on the same subnet. In addition to logging, the mirroring element can be used to create a hub, which is needed for experimenting with many intrusion detection systems such as Manhunt from Symantec. More details on this utility can be found in [5]

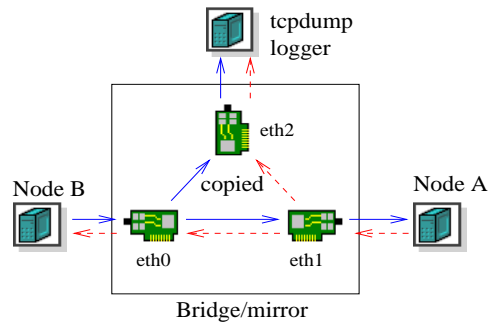


Fig. 4. Link monitor architecture.

#### C. Packet Generation

In order to benchmark the performance of systems on the testbed, we developed a highly flexible packet generation utility. The utility is capable of variable packet rates by utilizing UNIX real-time timers. In addition to supporting different packet generation rates, the tool can also generate varying packet sizes and supports ICMP, UDP, and TCP packet headers. We have also developed a pulsing mode for this tool, which can simulate highly periodic traffic.

### V. CONCLUSIONS

This paper has described the three required types of tools that we have developed in order to facilitate experimentation on emulation testbeds. Our control and measurement tools are also suitable for any laboratory test network, as they are independent of the emulation environment. More details about our tools can be found on our web page <http://www.cs.purdue.edu/~fahmy/software/emist/> and in [4], [5].

## REFERENCES

- [1] Gtnets. <http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/>.
- [2] Traceroute.org. <http://www.traceroute.org>.
- [3] The worlds leading network modeling and simulation environment. <http://www.opnet.com/products/modeler/home.html>, 2005.
- [4] R. Chertov, S. Fahmy, and N. Shroff. Emulation versus simulation: A case study of tcp-targeted denial of service attacks. In *Proceedings of the 2nd International IEEE CreateNet Conference on Tesbeds and Research Infrastructures TridentCom, 2006*, February 2006.
- [5] Roman Chertov. Performance of a software link monitor. <http://www.cs.purdue.edu/homes/rchertov/reports/click.pdf>, 2005.
- [6] L. Gao. On inferring autonomous system relationships in the internet. In *Proc. IEEE Global Internet Symposium*, November 2000.
- [7] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.
- [8] MOSES Project. iSSF and iSSFNet network simulators. <http://www.linklings.net/MOSES/?page=software>, 2005.
- [9] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with rocketfuel. In *Proceedings of ACM SIGCOMM*, 2002.
- [10] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A public Internet measurement facility. In *Proceedings of the 4th USENIX Symposium on Internet technology and Systems*, 2002.
- [11] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topology generators: Degree-based vs. structural. In *Proceedings of ACM SIGCOMM*, 2002.
- [12] UCB/LBNL/VINT groups. UCB/LBNL/VINT Network Simulator. <http://www.isi.edu/nsnam/ns/>, 2005.
- [13] F. Wang and L. Gao. On inferring and characterizing internet routing policies. In *Proc. Internet Measurement Conference (Miami, FL)*, October 2003.
- [14] E. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internet-work. In *Proc. of IEEE INFOCOM*, volume 2, pages 594 –602, March 1996.