Emulation Platform for Network Wide Traffic Sampling and Monitoring

Amir Krifa, Imed Lassoued, Chadi Barakat

Emulation Platform for Network Wide Traffic Sampling and Monitoring *

Amir Krifa, Imed Lassoued, Chadi Barakat
INRIA Sophia Antipolis - EPI Planète - France
Email: {Amir.Krifa,Imed.Lassoued,Chadi.Barakat}@sophia.inria.fr

ABSTRACT

It is of utmost importance for the network research community to have access to tools and testbeds to explore future directions for Internet traffic monitoring and engineering. Although many experimental solutions exist today, they tend to be highly specialized or to have a limited availability and openness. Through this work, we outline the monitoring capabilities limitations of these facilities and we present our emulation platform for network wide traffic monitoring as an answer to these limitations. Our platform presents a new approach for the emulation of Internet traffic and for its monitoring across the different routers. Through our solution, we put at the disposal of users a real traffic emulation service coupled to a set of libraries and tools capable of Cisco NetFlow data export and collection, the overall destined to run advanced applications for network wide traffic monitoring and optimization.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.10 [**Software Engineering**]: Design

General Terms

Algorithms, Design, Experimentation

Keywords

Traffic Emulation, Monitoring, Sampling

1. INTRODUCTION

For the purpose of testing new applications and protocols related to traffic monitoring and traffic engineering, the network research community has a persistent demand of either having a real-time control on real measurement points, or getting collected traces from these points together with the related network topology. When available, and even in the absence of real-time control, network-wide traces can be played a posteriori to simulate as close as possible the real environment. The problem is that these needs are most of the time not satisfied by ISP(s) for privacy and security reasons. Thus and as an intermediate solution, researchers make use of either network emulators coupled with synthetic traffic generators [2], or simply limit their research to parsing and studying traffic traces collected on specific links. Choosing one of these solutions depends on a set of factors namely the available information, the context of the algorithm to evaluate, the solution realism, the facilities provided by the solution in terms of monitoring and finally its extensibility.

As other researchers, we are also facing situations where access to network-wide traces and a control on measurements points are required. In particular, we are seeking a network-wide traffic monitoring platform that allows us to control the monitoring tools inside each router and to report measurement results to a central unit for further analysis. Such features are useful when studying several applications such as distributed anomaly detection, traffic engineering, and network wide optimization of monitoring tools. Unfortunately, such a platform is not available for researchers that do not operate real networks. One solution is to play synthetic traffic inside network simulators. Even though it is an interesting option, we discard it in our research for its lack of realism. Instead, we focus on the development of an emulation platform where real monitoring tools run inside virtual routers, and where generated traffic is faithful to the one in real ISP networks. Hence, the main question we try to solve becomes: starting from a *partial* set of collected real traces and given a network topology, how to build a platform of virtual routers respecting this topology, and how to dispatch and play over it the available traces while providing to the end-user remote controllable traffic monitoring capabilities for each router?

In this paper, we present the platform we developed as an answer to the latter question. Our platform presents a new approach for the emulation of Internet traffic and for its monitoring across the different routers. In its current version, the traffic is sampled at the packet level in each router of the platform, then monitored at the flow level[1]. We put at the disposal of users real traffic emulation facilities coupled to a set of li-

*This research work is partially funded by the European Commission through the ECODE project (INFSO-ICT-223936) of the European Seventh Framework Programme (FP7). Our platform: http://planete.inria.fr/NWTSM/

IWCMC'10, June 28 — July 2, 2010, Caen, France.

[1]A flow is a set of packets sharing common fields in their headers. The most common and basic definition is the 5-tuple one where packets of a flow share the same source and destination address, port number and transport protocol.

braries and tools capable of Cisco NetFlow[2] data export, collection and analysis. Our aim is to enable running and evaluating advanced applications for network wide traffic monitoring and optimization. The development of such applications is out of the scope of this research. We believe that the framework we are proposing can play a significant role in the systematic evaluation and experimentation of these applications' algorithms. Among the direct candidates figure algorithms for traffic engineering and distributed anomaly detection. Furthermore, methods for placing monitors, sampling traffic, coordinating monitors, and inverting sampling traffic will find in our platform a valuable tool for experimentation. This paper describes the platform we propose and is structured as follows. After presenting related work in Section 2, we describe in Section 3 the issues related to our platform design. In Section 4, we present the results of our validation tests and analyze them. Then, we conclude our work and present the future development direction of our platform in Section 5.

2. RELATED WORK

Several recent works are interested in the evaluation of traffic monitoring solutions. The majority of these works deal with the understanding of the content of the traffic on some network link. They use for their evaluation packet level traces collected by network operators or by researchers themselves, then made available after anonymization. The traces of the Japanese MAWI project [8] are a typical example that we use in this work. Packet level traces have been used for several purposes such as studying the statistical properties of Internet traffic [5], detecting anomalies and attacks [17], validating efficient methods for application identification and classification [1], and studying the accuracy of estimating traffic statistics from a stream of sampled packets [6]. Works in this list, which is far from being exhaustive, all share the feature of only requiring the availability of traces. Clearly, the more of these traces are available, the stronger is the validation.

On the other hand, we find studies that require the existence of network-wide data and access to routers and monitors. Unfortunately, those studies are more challenging since their validation requirements cannot be satisfied, except by network operators who possess such data and monitors. Therefore, the most common approach is to resort to simulations or emulations where synthetic traffic is played across the studied ISP topology. In [14], the authors follow this approach to optimize the placement of monitors and the sampling rates inside routers. In [12], Sekar et al. develop their own synthetic traffic generator tool; they use Emulab [3] for setting the test network topology and YAF [4] as a monitoring tool to capture flow information. The existence of network-wide real data would help to strengthen the validation of these solutions even further. Such data has helped Duffield et al. in [4] to propose a solution for estimating network-level flow rates from measurements taken at multiple routers. They use sampled NetFlow records gathered from two routers in a major ISP network. Lakhina et al. [7] have used NetFlow data collected from the PoP routers of a tier-1 ISP network to detect network-wide traffic anomalies and classify them. Generally, the collected data is parsed offline with appropriate scripts implementing the proposed algorithms. It is also used to calibrate models for the generation of synthetic traffic. Unfortunately, real access to monitors in order to analyze the reports they send and to apply the proposed algorithms on the fly is still hard if not impossible to obtain by the research community.

[2]Cisco Netflow, `http://www.cisco.com/`

[3]Emulab: Network Emulation Testbed Home, `http://www.emulab.net/`

[4]YAF: Yet Another Flowmeter, `http://tools.netsa.cert.org/yaf`

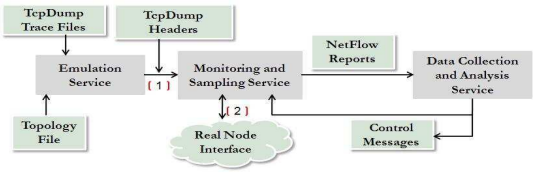

Figure 1: Our Platform Architecture

Our platform provides a tool for researchers to play real TcpDump packet traces [15] over a target network topology and to control the monitors installed inside routers. These monitors sample and capture the traffic, then send flow-level reports about it in real time to a central collector. The key feature of our platform is that it splits the available packet-level traces across the emulated topology without resorting to synthetic models. This guarantees realism while providing enough freedom in defining the way the real trace is split and the network topology is formed. Routers are virtual entities of very low complexity, which allow the platform to scale to large networks as long as rich packet-level traces exist. Users can control the way the traffic is sampled and monitored, and can use the real time reports sent by monitors to feed their own applications. The applications they develop over our platform can be run later as they are over real networks.

3. PLATFORM ARCHITECTURE

The objective of this work is to build a real environment for flow-level monitoring inside ISP networks starting from traffic emulation to traffic sampling and monitoring. We reproduce and approximate this real environment through our platform described in Figure 1. The latter depicts the interactions between the three main components of our architecture: the emulation service, the flow monitoring and sampling service, and the flow data collection and analysis service.

A common usage scenario of our platform can be summarized as follows. The user starts by supplying the emulation service with two configuration files describing respectively the list of packet-level TcpDump traces to play and the network topology. Once done, the user runs the three services of the platform either in the same machine or in three different machines for better performance [5]. The emulation service creates the virtual routers and network links to emulate the described topology, then dispatches the set of IP addresses available in the traces over the emulated topology and plays the packets accordingly. The monitoring and sampling service installed in each router allows sampling the traffic at the packet level, before constructing flows and exporting flow reports to the data collection and analysis service. Within the latter service, the user can plug and run any advanced traffic analysis algorithm. Next, we describe the different services with further details.

3.1 Traffic Emulation Service

[5]For a better scalability and as part of our future work, we intend to enhance the platform in such a way that users can emulate different sets of routers over different machines.

The traffic emulation service provides means to describe a given network topology. The emulated network is fed with a real traffic captured on some high speed link in a backbone transit network, before being dispatched and played over the emulated topology. Figure 2(b) depicts an example of a simple topology that users can describe. We provide a highly flexible configuration methodology via XML files through which users can describe their emulated network. In particular, they can describe the different stub ASes and their associated weights[6], the list of routers (interfaces, IP addresses), the set of characteristics of the different links as well as the list of monitors deployed inside routers. Once the emulated topology is available, our emulation service looks for IP prefixes within the available TcpDump traces and dispatches them over the stub ASes according to their weights. The packets of these traces are then played over the routers of the topology respecting their timestamps to reproduce the network wide packet-level traffic we are looking for.

Figure 2(a) shows the processing flow within the emulation service. As the traffic loader module reads packets from the TcpDump trace using the Pcap library [15], the dispatching module associates them online to the right AS based, on one hand, on the list of weights the user assigns to the different stub ASes, and on another hand on the prefix length specified by the user for the dispatching. The default dispatching method we are implementing is the *weighted random*. Suppose we have 23 ASes to whom we associate different weights. We subdivide the interval $[0, 100]$ into 23 consecutive bins, where each bin represents the weight of an AS. Suppose further that the user chooses the prefix length to dispatch as being /16. Then, each time a new prefix of the same length appears while reading the raw trace, a random number is selected within the interval $[0, 100]$ using a uniform distribution. This selected number points to the AS to which the new prefix is to be associated. All subsequent packets having the same prefix as source or destination are associated to the same AS [7]. Once loaded packets are dispatched, they are scheduled and played within the routers' emulation module. As a final step, for each router of the transit network selected as monitor, the data forwarding module enables the monitoring and sampling service over the interfaces of this router.

To scale up the experimentation to large scenarios, we parallelize the maximum number of tasks within the emulation service, namely those required for trace parsing, IP prefix mapping and packet playing. The objective is to profit from all the CPU power provided by the host machine and to prevent excess latency during packet scheduling. So, all the modules described in Figure 2(a) run in parallel. Furthermore, we resort to massive dynamic memory development to minimize the emulator memory footprint. Indeed, we profit from the parallelism that we have introduced to parse and play large TcpDump binary files without having to load them entirely into the memory. Instead, as loaded packets are played inside routers through the emulation module and consumed by the data forwarding module, new ones are loaded via the traffic loader module. The amount of packets to load at once is dynamically adjusted to limit the latency introduced during packet scheduling.

[6]ASes connected to the emulated network are supposed to generate different amounts of traffic with respect to the whole network traffic.

[7]Depending on users' needs, one could imagine other dispatching methods as well.

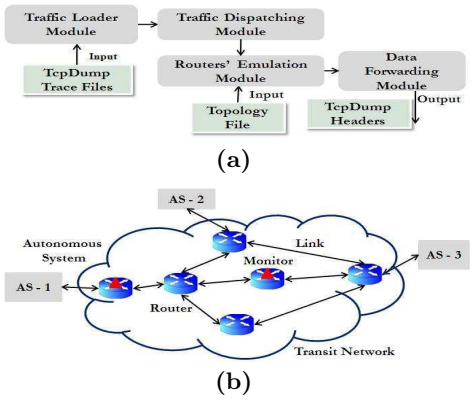

Figure 2: Traffic Emulation Service

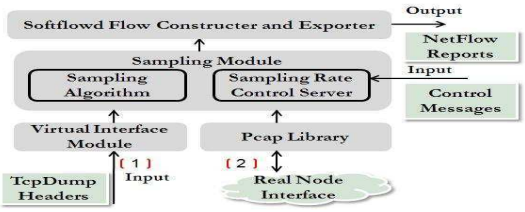

Figure 3: Monitoring and Sampling Service

In terms of extensibility, new traffic control and monitoring methods can be added to our platform without major changes to the emulation service design. In addition, users interested in studying the performance of different routing algorithms can plug their own routing modules provided they maintain the same programming interface as the default one. For the moment, we are only supporting the shortest path routing protocol based on the Dijkstra algorithm [3]. Routes are static and are set up via the XML configuration file.

3.2 Traffic Monitoring and Sampling Service

We have designed and developed the traffic monitoring and sampling component as an extension of Softflowd [13]. Indeed, Softflowd is a flow-based open source network traffic analyzer capable of Cisco NetFlow data export. Softflowd does not include support for packet sampling neither fixed nor adaptive. Our extensions to Softflowd add this sampling functionality, which is essential for monitoring scalability. We also allow the integration of this tool over virtual nodes fed by our emulated traffic, while being able to run over real routers by sniffing packets directly on their interfaces. The last functionality enables users to first run and evaluate their sampling and monitoring methods on a set of nodes within our controlled environment. Then, they use the same methods within real routers without any extra development effort. Next, we describe these extensions through two typical usage scenarios.

- ***Experimenting with emulated routers:*** The monitoring and sampling service communicates with the traffic emulation service via the virtual interface module described in Figure 3. It requires specifying the port number on which the virtual interface module will listen to incoming packets (per-monitor TcpDump headers) from the traffic emulator. Received packets are sampled via the sampling module and forwarded (if chosen) to the Softflowd flow constructer and exporter module, which

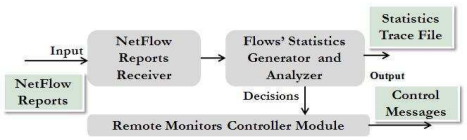

Figure 4: Data Collection Service

takes in charge the construction of the flows and the compilation of reports to be sent to the data collection and analysis service described in Figure 1.

- ***Experimenting with real routers:*** Our monitoring and sampling service is also designed to be plugged directly on a real router interface. In this case, it captures real traffic promiscuously using the Pcap library [15] as described in Figure 3. The sampling module will likely place additional load on hosts or gateways on which it runs. Our implementation has been designed to minimize this load as much as possible. Indeed, in order to decide either to consider the packet being read or to reject it, the sampling module makes a decision each time the Pcap library returns a handle to a new packet before it is being loaded into the memory. If the sampling algorithm decides to capture the packet, the entire packet is loaded and the maintained flow list is updated via the Softflowd flow constructer module, otherwise the packet is simply discarded.

As described in Figure 3, the sampling module encloses a sampling algorithm as a core and a sampling rate control server. The default sampling algorithm we are providing is the following: [8] if a user chooses a sampling rate of A/B (A packets among B packets, $A \leq B$, $B > 0$, $A \geq 0$), then every B packets, the sampling module generates randomly a set S of A numbers within the interval $[1, B]$. Packets with numbers outside the set S are rejected and only the remaining packets are considered for 5-tuple flow construction. Generated flows are then encapsulated within NetFlow reports and are exported via the Softflowd flow constructer and exporter module. Concerning the sampling rate control server, it enables users to change remotely the sampling rate of a given monitor whether it is in a real network or within the traffic emulation service. A remote monitor controller, which we will describe later, proceeds to change the local sampling rate to each monitor. This remote control functionality allows users to control and change online the sampling rate of one or multiple monitors, or simply offline from one experiment to another.

3.3 Data Collection and Analysis Service

Figure 4 depicts the principal modules of the data collection and analysis service. We design and develop the data collection and analysis service starting from the functionalities proposed by Flowd [13]. As described in Figure 4, we enclose the Flowd capabilities in the NetFlow reports receiver module and develop around it other modules namely the flow statistics generator and analyzer, and the remote monitor controller. Depending on their needs, users can easily customize this module to infer different statistics. For example, one can implement anomaly detection based on the collected NetFlow reports or introduce quality of service algorithms that improve traffic routing as a function of the monitored network status. Furthermore, users can decide to change the sampling rate of one or more monitors to improve the accuracy of the algorithms implemented at the analyzer, either offline or online. For this, they have to use the remote monitor controller. The latter module provides an API to send control messages to a monitor specific control server, which proceeds to modify the sampling rate.

[8]Other sampling algorithms can be easily deployed.

4. VALIDATION AND DEMONSTRATION

To illustrate the operation of our platform, we emulate traffic over a backbone network with several stub ASes. The traffic is originated from a tier-1 transit link and is dispatched over the emulated topology according to some weight associated to each stub AS. We show monitoring results on the number of sampled flows and the number of packets across the different edge routers to check whether they follow the access network weights (or traffic matrix) predefined in our experiment configuration file.

4.1 Test Environment

As an example of an emulated topology, we take the one of the GEANT tier-1 backbone [18]. GEANT is a pan-European transit network that connects Europe's national research and education ASes via 23 routers. The traces to play over this topology are collected at a trans-pacific link by the Japanese MAWI working group [8]. Those traffic traces are made by TcpDump, and then, IP addresses in the traces are scrambled by a modified version of Tcpdpriv [16]. For traffic dispatching over the different access networks (European countries in the case of GEANT), we associate different weights to the 23 stub ASes based on the volume of the traffic they are supposed to generate in reality. We infer these weights from the populations of the countries represented by these ASes and the capacities of the links that connect them to their respective access routers in GEANT. We run the emulation service as well as the monitoring and sampling service in the same machine while keeping the traffic collection and analysis service in a different machine (connected via an Ethernet network). The traffic collection and analysis service is supposed to be a centralized component that collects NetFlow reports sent by the monitoring and sampling service installed in each of the 23 routers. The latter connects to each of the 23 emulated routers via its virtual interface module. Our aim is to obtain a traffic across the emulated topology that respects the sampling rate in routers and the weights associated to stub ASes. However, we do not claim that this traffic is representative of GEANT traffic, we are only using the GEANT topology as a network example.

4.2 Results

Figure 5(a) depicts the evolution of the network-wide monitored number of 5-tuple flows as a function of the sampling rate in routers. In each experiment, we take a different sampling rate while maintaining it constant for all routers. As expected, the number of 5-tuple flows decreases linearly as we decrease the sampling rate in all the access routers. Indeed, if S is the size of a given flow, then the probability that this flow is sampled given a sampling rate p is equal to $1 - (1 - p)^S$, which can be approximated by $p.S$ for small p. The number of sampled flows N_p can then be approximated by $p.E[S].N$, where N is the total number of original flows. The latter quantity decreases linearly with the sampling rate.

Next, we look at the effectiveness of the emulation service and especially at its dispatching mechanism. We try to answer the following question: does each AS generate as much traffic as the weight associated to it? Towards that, we start by plot-

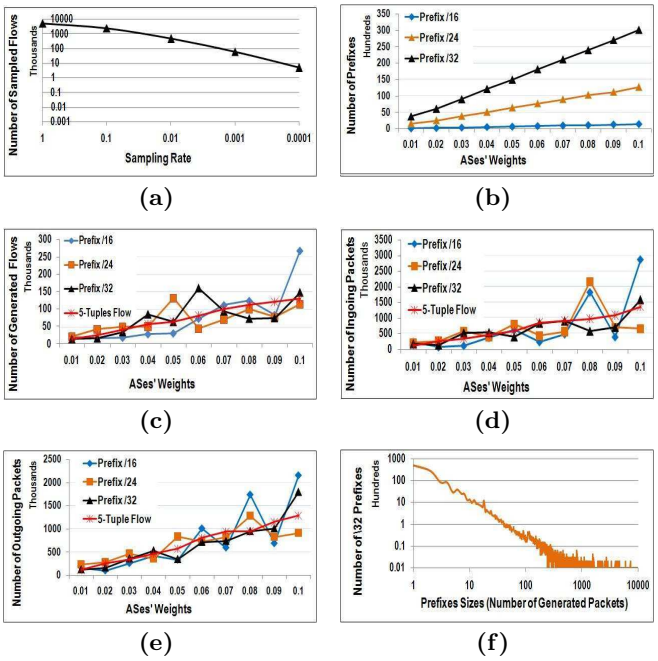

Figure 5: Validation Results

ting in Figure 5(b) the number of prefixes per AS function of the weights associated to ASes. The resulting curves remain linear for different prefix lengths. So, we conclude that our emulator dispatches the prefixes to ASes without any bias. Then, we look at the number of generated flows, ingoing and outgoing packets per AS in Figures 5(c), 5(d) and 5(e), respectively. We notice that for the three considered prefix lengths (/16, /24 and /32), the number of generated flows as well as the number of ingoing/outgoing packets scale with the AS weights but do not fit a perfect line. Nevertheless, the fitting improves when the prefix granularity becomes finer. Indeed, the presence of prefixes of very large but different volumes causes such deviations in the traffic; the coarser the prefix the more important this phenomenon. We illustrate it on prefixes of length /32 (IP addresses), for which we plot the distribution of their sizes (in packets) on a log-log scale in Figure 5(f). Clearly, there is a power-law behaviour leading to very large prefixes compared to the average prefix size (these are servers, heavy users, etc). To improve further the fit, one can run the dispatching at another finer granularity, the 5-tuple level. This finer granularity enables our dispatching algorithm to split a big set of 5-tuple flows generated by (or destined to) a single /32 prefix to different ASes. As we can see in Figures 5(c), 5(d) and 5(e), the number of generated flows and ingoing/outgoing packets now better fits a line. Even though it preserves the notion of connections, the 5-tuple dispatching still has the problem of altering the pattern of activity of servers and end hosts. The choice of the best prefix length should be decided by this trade-off established between traffic realism and AS weight respect.

5. CONCLUSIONS

In this paper, we describe our emulation platform for network wide traffic sampling and monitoring. The architecture that we propose contains three main components: an emulation service, a monitoring and sampling service, and a data collection and analysis service. The design of these services takes into consideration the constraints of environment conservation, scalability and extensibility. Our platform offers a complete set of features towards the development and evaluation of solutions for network monitoring and management. Namely, it offers the possibility to reproduce real backbone network topology, to monitor and sample the packets being forwarded in a given router and finally to analyze the collected flows. Our platform allows users to remotely tune the sampling rate of a given monitor to balance between accuracy and overhead. As future work, we intend to leverage this platform in a network-wide adaptive monitoring infrastructure that optimally configures monitors as a function of the targeted monitoring task. The development of distributed anomaly detection algorithms is another perspective of this research.

6. REFERENCES

[1] L. Bernaille, R. Teixeira, and K. Salamatian, "Early Application Identification", in proceedings of the 2nd CoNEXT Conference, Lisboa, Portugal, Dec 2006.

[2] D. Brauckhoff, A. Wagner, M. May, "FLAME: A Flow-Level Anomaly Modeling Engine;" in proceedings of CSET, 2008.

[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, "Introduction to Algorithms," MIT Press and McGraw-Hill, 2001.

[4] Duffield, Nick and Lund, Carsten and Thorup, Mikkel, "Optimal combination of sampled network measurements," Proceedings of IMC, Berkeley, CA, 2005.

[5] A. Feldmann, A. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," in proceedings of ACM SIGCOMMŠ99, Vancouver, Sep 1999.

[6] N. Hohn and D. Veitch, "Inverting sampled traffic," IEEE/ACM Trans. on Networking, 14(1):68Ű80, 2006.

[7] A. Lakhina, M. Crovella and C. Diot., "Mining Anomalies Using Traffic Feature Distributions," in proceedings of ACM SIGCOMM 2005.

[8] MAWI Working Group Traffic Archive, `http://tracer.csl.sony.co.jp/mawi/`.

[9] The Network Simulator (NS-3), `http://www.nsnam.org/`.

[10] PlanetLab, `http://www.planet-lab.org/`.

[11] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," ACM CCR, 1997.

[12] V. Sekar, M. Reiter, W. Willinger, H. Zhang, R. Kompella, and D. Andersen, "cSamp: A System for Network-Wide Flow Monitoring," in proceedings of the 5th USENIX NSDI, San Francisco, 2008.

[13] Softflowd and Flowd, `http://www.mindrot.org/projects/flowd/`.

[14] K. Suh, Y. Guo, J. Kurose, and D. Towsley, "Locating network monitors: Complexity, heuristics and coverage," in proceedings of IEEE INFOCOM, Mar 2005.

[15] TcpDump/LibPcap, `http://www.tcpdump.org/`.

[16] Tcpdpriv, `http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html/`.

[17] M. Thottan and C. Ji, "Anomaly Detection in IP Networks," IEEE Trans. Signal Processing, vol. 51, issue: 8, pp. 2191 -2204, Aug 2003.

[18] XML description of the GEANT topology, `http://planete.inria.fr/NWTSM`.