An Open Architecture for Distributed IP Traffic Analysis (DITA)

Cristian Morariu, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IFI, University of Zürich
Binzmühlestrasse 14, CH—8050 Zürich, Switzerland
E-mail: [morariu|stiller]@ifi.uzh.ch

***Abstract* — This thesis investigated how performance of today's IP traffic metering and analysis applications can be improved by moving from a centralized, high-performance infrastructure, which executes these tasks, to distributed mechanisms, which combine available resources of multiple devices. The results achieved show that distributed IP traffic metering and analysis leverages bottleneck problems. The distributed IP traffic approach DITA does not solve all problems of handling such large amounts of data in very short time by itself, but proposes an orthogonal approach to existing solutions. DITA revelas that combining distributed IP traffic metering and analysis reaches better and higher performance sampling and aggregation mechanisms, which do provide a very flexible and the open solution to analyzing IP traffic in future high-speed networks. This has been achieved by the facts that all mechanisms designed for DITA — and their prototypical implementations — are based on standard protocols and open-source technologies. DITA determines the first approach to distributed IP traffic metering and analysis known today, which (a) addresses the different bottlenecks of traffic analysis in a generic way, and (b) is self-organizing, offering a scalable solution to regular traffic increases.**

I. Introduction

Since the first days of the Internet the IP (Internet Protocol) traffic carried by network operators increased year after year. This was mainly caused by a continuous growth in the number of users having Internet access, combined with an increase in services that those users have access to using an IP infrastructure. Traditional telecommunication services that had their dedicated infrastructures (such as telephony, television) are in a process of gradually switching to IP. Two additional causes, which have lead to the increase of network traffic are a constant need of users to have access to higher quality services, and, the pervasiveness of modern mobile devices, which allow users to be connected to the Internet anytime and from almost anywhere. Different studies of the evolution of Internet traffic show that on average, during the last decade, Internet traffic increased between 50% to 100% every year. By 2012 the total Internet traffic carried by Internet providers is estimated to be about 75 times higher than the total traffic carried in 2002 [1].

In order to properly address the challenges of high speed traffic monitoring, key problems in *IP Traffic Monitoring* need to be identified; centralized traffic monitoring and analysis architectures experience bottlenecks at different stages in the monitoring pipeline (*e.g.* during metering, exporting, or analysis). The metering process quickly becomes overloaded if the time required to process a single packet exceeds the interarrival time between two consecutive packets. In case of counting the amount of traffic (number of bytes and packets) observed on a network interface, the processing of a single packet requires updating several counters (such as a packet counter, a byte counter, unicast/multicast counters, etc.) mapped to the interface on which the packet was observed. Although each packet might change tens of counters, such operations can be done at line speed, even in case of high packet rates, as these counters can be kept in fast memories, such as Static Random Access Memory (SRAM), or even registers or line processor caches. In case of flow accounting data about active IP flows is stored in a flow cache. Each packet triggers a lookup into the flow cache to retrieve the flow record which needs to be updated. Even with efficient flow cache management algorithms (*e.g.*, hash tables) such a process triggers multiple accesses to the main memory, where the flow cache is stored. In case of high-packet rates the time required to find a corresponding flow record entry often exceeds the packet interarrival time. As a result, not all packets can be processed [6].

Another component that often experiences bottlenecks is the data exporting process from the metering point [2]. If millions of different active flows exist in the network the flow cache memory fills very quickly and flow records need to be exported in order to create space for new ones. In case of attacks, or a small flow cache memory, it is often the case that most of the observed packets create new flow records. The rate of creating new flow records can easily exceed the rate at which the exporting process can export this data, which leads to a bottleneck caused by the exporting process.

Besides metering and exporting process, problems also appear at a data collector or during analysis. Often a network operator collects metering data from multiple observation points in his network. If all data are collected by a centralized collector it may cause bottlenecks on the network link, which aggregates all data exported. In addition, if the collected records need to be stored in a persistent memory, the rate of incoming data could exceed the rate of writing in the persistent memory. An analysis application may experience a bottleneck similar to the metering process, when the rate of incoming metering records exceeds the rate at which these records are processed.

As it was observed during the recent years solving the problem of a bottleneck in the monitoring and analysis pipeline of a large network only moves the problem to another component [2]. This thesis adresses the bottleneck problem of traffic monitoring and analysis in high-speed networks in a generic

way which includes mechanisms to alleviate all those bottlenecks described earlier.

The main goal of DITA is to develop an open and generic architecture which enables distributed traffic monitoring and analysis. Starting from the above observations those challenging aspects of traffic metering, monitoring, and accounting in high speed networks are investigated, and a new architectural design to handle those problems in a distributed system is proposed.

Thus, the first problem which this thesis investigates is ***what is the effect of distributed IP traffic analysis?*** A generic architecture for IP traffic analysis, independent of the analysis applications runing ontop of it, is important, as it is the basis for future scalable traffic analysis infrastructures. Such a system allows network operators to scale up their traffic analysis infrastructure by adding new machines, rather than upgrading or changing it completely as it is typically the case nowadays. The second problem that this thesis investigates is ***how can the performance of existing traffic monitoring applications which run on off-the-shelve PCs be improved?*** It was observed that at high packet rates packet-capture libraries cause the operating system to spend most of its resources on capturing packets, while leaving less resources for the monitoring application, thus, causing an overload of the system, which eventually leads to dropped packets. Finally, the third problem investigated is ***how to increase the granularity of IP metering data, so that an IP packet can be mapped to an individual user or even a process and application***? The traditional way to address this problem is to assume that an IP address is used by a single user at a time and have a mapping between IP address-to-user mapping all times. In case of applications, the straight-forward approach is to map ports to applications. In case of users the problem arises when the end systems are multi-user capable and several users run at the same time network applications , while in case of application accounting ports are not reliable anymore in concluding the application for a given packet, as more and more applications use the Hypertext Transfer Protocol (HTTP) [7] for exchanging data (and often port 80).

II. Distribution Model

In order to address those bottlenecks outlined above in an integrated way, and to avoid the shortcomings of existing distributed approaches to IP traffic analysis, the generic model DITA (Distributed IP Traffic Analysis), and its attached architecture were developed. This model defines characteristics and requirements of distributed traffic analysis and outlines its main building blocks.

A. Centralized vs. Distributed Traffic Analysis

A typical deployment for a centralized traffic analysis application is depicted in Figure 1. A centralized collector received flow records from a set of exporters (*e.g.* routers) in the network. Upon the receipt of these records additional analysis applications use them for different purposes, such as accounting, charging, intrusion detection, network monitoring, etc. The traffic increase experienced by the network operator also translates into an increased amount of flow records that need to be handled by the collector and analysis applications. This leads to a situation in which an existing collector does not have sufficient resources to handle the data at the desired rate, thus, the central collector needs to be replaced with a new, more powerful, but also more expensive machine. Eventually, this new machine will have similar problems in future and will have to be replaced again.

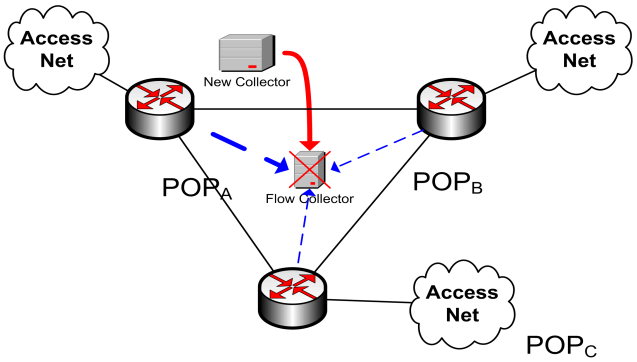

Figure 1: Centralized Flow Collector Replacement

A directly distributed approach, such as just adding a new collector and configuring some of the routers to forward their flow records to this new collector, is not feasible, as often correlations between flow records received from multiple sources are required. Such correlations include detection of duplicated flow records in order to delete redundant data, or calculation of some network parameters based on traffic observed at multiple points. In case of two independent collectors an external component is required to perform such correlations, therefore, the bottleneck is not eliminated, but pushed to another component.

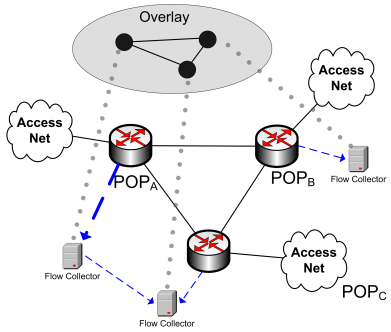

Figure 2: SCRIPT Approach to increased IP metering data

The new DITA approach is summarized in Figure 2. Multiple data collectors form a self-organizing overlay which includes nodes that perform traffic analysis. Routers can choose any of the existing collectors to export their data to, while the overlay ensures that the exported data reaches the intended analysis applications. Using such an approach allows a network operator to address increases in traffic to be analyzed by adding new machines to the analysis overlay.

B. Scenarios for Distributed Traffic Monitoring

To extract key requirements for a distributed IP traffic moni-

toring seven scenarios grouped in three areas (Flow record analysis, high-speed metering, per-user IP accounting) have been selected to outline the basis of the design (cf. Table 1).

TABLE 1 Scenarios for Distributed Traffic Monitoring

Flow Record Analysis	High Speed Metering	Per-user Accounting
Data Retention	Packet Capture and Analysis on High-Speed Links	Billing for IP Traffic
Delay Measurements		User monitoring and abuse detection
Real-time Asymmetric Route Detection		Service Load Monitoring

1) Flow Record Storage and Analysis

The first set of scenarios deal with flow record processing. Three different scenarios are presented below: *Data Retention, Delay Measurement, and Real-Time Asymmetric Route Detection.* The reason for choosing several scenarios was to cover analysis applications that run off-line and require stored data, as well as applications that are real-time sensitive. The Data Retention scenario [11] is common to most network operators, as it deals with storage and retrieval of IP metering data over a longer time. Legislation was enforced in different countries [3], [4] forcing network operators to keep traces of the traffic created by their users. Monitoring the delay is a key element for any network operator that wants to offer high quality services which are defined by a Service Level Agreement (SLA). The third scenario — Asymmetric Route Detection (ASR) — was chosen in order to present a real-time sensitive traffic monitoring application which needs to process huge amount of data in little time.

Shortcomings of Centralized Solutions

The major disadvantages of centralized solutions which were observed in the above investigated scenarios can be summarized as different bottlenecks due to:

- Incoming IPFIX data arrives at a rate higher than the maximum write rate of the hard disk or storage device;
- The network link bandwidth of the centralized collector is not sufficient for the aggregated IPFIX streams from all exporters; and
- In case of real-time processing, required at collector's side, processing time of an IPFIX record is higher than the records' inter-arrival time.

2) Packet Capture and Analysis on High Speed Links

This scenario stems from real problems encountered in many network monitoring research labs by researchers doing packet inspection at high packet rates [10]. Performing software-based packet inspection at high packet rates is very difficult, as the processing time of a single observed packet easily exceeds the packet inter-arrival time on that link. In such situations, sampling is used in order to reduce the number of inspected packets. Even less complex measurements, such as IP Flow accounting use sampling rates of 1/1000 on multi-gigabit links.

One of the main problems in capturing and analyzing packets on high-speed links is the very short time that can be spent handling a single packet. As shown in [12] 44% of the IP traffic observed on an Internet Service Provider (ISP) in today's Internet is made of packets with sizes between 40 and 100 byte. Assuming a 10 Gbps Ethernet link fully loaded with 64 byte packets — which are very common in voice over IP (VoIP) applications — this would translate into approximately 20 million packets per second or approximately 50 ns for handling a single packet by the packet inspection node. Capturing packets requires high-performance memory typically exceeding the speed of DRAM (Dynamic Random Access Memory) memory existing in standard PCs. Dedicated capturing cards, such as Endace's DAG cards [5], make use of the faster and more expensive SRAM (Static Random Access Memory) memory and are able to capture at those high packet rates, but they typically come at high prices.

Shortcomings of Centralized Solutions

Due to the short interarrival time between consecutive packets flowing through a high speed link software-based packet processing applications cannot process every single packet and often process only a sample of the packets observed on the link. In case of network intrusions the packets responsible for such attacks may be among the unsampled packets. A distributed system can address this problem by enabling distributed processing of packet data.

3) Per-user IP Traffic Accounting in a Large Enterprise Network

Internet Service Providers often perform traffic accounting in order to charge their clients according to the data volume they transferred over a period of time. Such a charge does not necessarily consist of monetary units, but could be a penalty or incentive in order to reduce or increase a user's traffic (for example, some operators offer flat-rates subscriptions which include a drastic bandwidth limitation if a certain download limit in a month is exceeded). Such type of accounting process is easily doable if it is assumed that a user can be uniquely identified with an IP address at a given point in time. All that a network operator needs to do is to correlate its IP metering data with the information about which user is assigned each of those addresses. However, there are network scenarios such as multi-user operating systems in which such an assumption does not hold. On such systems multiple users might have their applications running at the same time, each generating network traffic. Using a traditional IP accounting mechanism, all that a network administrator could see is how much traffic such a system generated, but not from which applications that traffic originated and which user started each of those applications. Such a scenario is easily encountered in grid environments, where multiple users share grid resources in parallel, or in enterprise networks, where all the users of a company (or uni-

versity) have access on any system in that network using a personal username and a password. The following three scenarios motivate the need for a distributed user-based IP accounting.

Shortcomings of Centralized Solutions

As traditional IP traffic accounting systems rely on measurement points located in network routers or switches which meter the IP traffic in the network based on the IP addresses in the IP header, they cannot map network traffic to a particular user or application. The only place where this information is accessible is in the end-node itself, which keeps a mapping between network sockets and the applications that created those sockets.

C. Requirements

Based on the discussion in the above scenarios, a set of requirements for distributed traffic monitoring and accounting are derived and summarized below:

R_1: Scalable Traffic Analysis without Sampling

R_2: Flexibility

R_3: Incremental Scalability

R_4: High Availability

R5: Based on Commoditty Components

R_6: Ability to detect originating end-user or processes in case of network abuse

R_7: Based on Open Standards

III. Model and Solution for the Distributed Traffic Metering and Analysis

To develop an integrated solution to the management problems discussed, a respective model for distributed traffic metering and analysis is designed. Figure 3 shows the distributed IP traffic monitoring and analysis model proposed by this thesis. It shows a layered architecture including a metering layer, a monitoring and analysis layer, and a presentation layer. The distributed metering layer includes one or more metering systems which are responsible with extracting the relevant data from the observed traffic. In order to cover also the user-based IP accounting problem this layer includes a model for general packet capture and processing, and another model for user-based IP traffic accounting.

The second layer shown in Figure 3 represents a distributed analysis system which enables traffic analysis applications to be deployed in a distributed environment. IP metering data is received from the metering system embedded in IPFIX records and uses internal mechanisms to forward this data to application instances that use it.

Finally, the third layer is a presentation system which allows a human administrator, or other external applications to visualize the results of the analysis process. A presentation system is dependant on the analysis application and usually has a different functionality and behavior in different analysis applications, thus, it is out of the scope of this thesis.

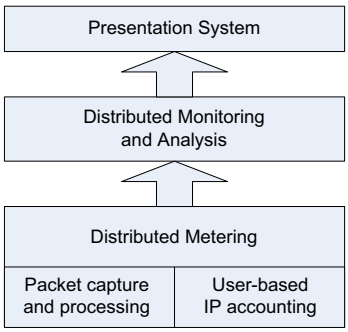

Figure 3: Distributed IP Traffic Monitoring and Analysis Model

Figure 4 shows the building blocks of a distributed traffic analysis system and their interrelations. In a network there are multiple *network components* (such as routers, switches, links, services, etc) that need to be monitored. The operation of these components is observed and measured by a *Meter*. One meter can measure more than a single component, for example it could measure traffic aggregated from several routers. At the same time a network component can be metered by multiple meters, for example one meter doing packet measurements, and a second doing flow measurements.

Another example of multiple meters serving a single network component is a distributed meter, having several meters, each monitoring the same network link. The metered data, once it is produced, needs to be sent to be exported to one or more *Data Collectors*. These data collectors perform limited pre-processing tasks, such as aggregation, anonymization, filtering, or encapsulation, which prepare the data to be used by traffic analysis applications, before feeding the received data to a *Traffic Processing Platform.*

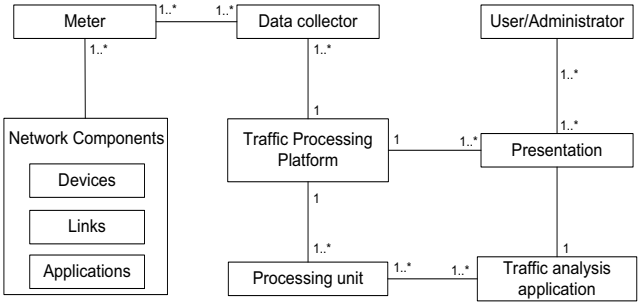

Figure 4: Distributed Traffic Analysis Components

The encapsulation process is of particular importance as its task is to switch the format of the received metered data (*e.g.* SNMP, NetFlow v5, Diameter, IPDR, proprietary protocols) to IPFIX which is used by the traffic processing platform. The traffic processing platform consists of one or more *Processing Units.* Each processing unit runs one or more *Traffic Analysis Applications.* It is the task of the traffic processing platform to feed each piece of metering data to the right analysis application instance.

The results of the traffic analysis applications are fed to a *Presentation* component which presents them to a *User* or *Administrator.* The presentation component also maintains a relation with the underlying traffic processing platform which allows it to access different traffic application instances.

IV. Evaluation

The evaluation of two of the main mechanisms developed in this thesis named SCRIPT [8] and DiCAP [9] are briefly summarized here. The main purpose of SCRIPT is to distribute IPFIX records to several machines according to rules required by an analysis application. This is achieved by organizing participating nodes in a P2P overlay and by using the P2P overlay information for distributing the IPFIX records. Using the API provided, applications can define routing functions according to their dedicated requirements.

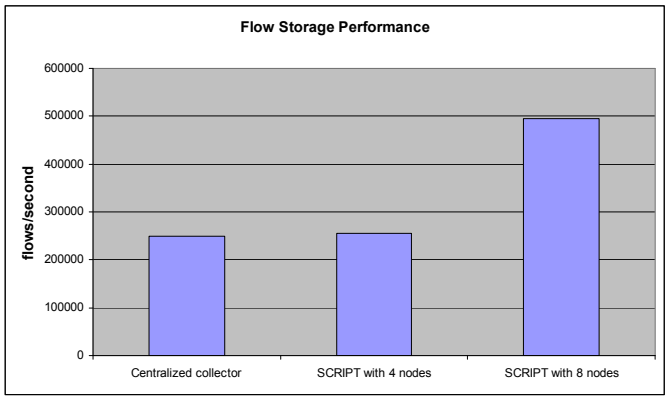

Figure 5: Flow Storage performance

Thus, the performance of the SCRIPT prototype is complex to be assessed, especially in comparison with other tools, since no such generic frameworks for distributed traffic data analysis exist. Therefore, the performance evaluation includes an evaluation of IPFIX records storage in a traditional, centralized collector, compared to the performance of a distributed collector built on top of SCRIPT. The tests were made using similar PCs with 3.6 GHz Intel processors, each having 4 GB memory. All tests have been performed in a switched Local Area Network, each PC having a 1Gbps network card. On the centralized collector the maximum rate of flow records that could be saved was 250,000 flows per second. Using SCRIPT running on 8 similar PCs in parallel a rate of 600,000 flows per second was achieved. In this evaluation, one stream of 150,000 flows per second was sent to 4 of the 8 nodes. Using only 4 nodes with SCRIPT the maximum flow rate that could be achieved in this prototype was 269,000 flows per second. These results are summarized in Figure 5.

DiCAP evaluation results are summarized in Figure 6. The figure shows that with traditional libpcap the maximum number of packets captured is around 50% for packet sizes of 512 Byte. Lower loss rates were expected at larger packet sizes as the packet rate for these is lower. It can be observed that with two capture nodes running libpcap in parallel using DiCAP the capture rate was doubled for small packets and was increased more than 5 times for packet sizes of 256 Byte. With just two capture nodes it was possible to capture all packets of 512 Byte. With four capture nodes running in parallel libpcap the capture rate for very small packets (40 Byte) increased tenfold while for 256 Byte packets the capture rate was 100%. Another test not shown in the figure shows that with five parallel capture nodes a capture rate of 100% can be achieved even for the 40 Byte packet sizes.

V. Summary and Conclusions

As outcome of the these investigations, this thesis proposes a generic model for distributed traffic metering and analysis named DITA. In order to evaluate the model several different mechanisms have been developed and presented here, which show that traffic metering and analysis can scale with the increase of traffic carried by backbone network links. The need for such a solution is obvious, as most network operators already use sampling and aggregation techniques in order to reduce the amount of data they meter, in order to meet the capabilities of their existing metering and analysis infrastructure. The mechanisms developed during this thesis could bring benefits to network providers, service providers, as well as end users. By using SCRIPT and DiCAP, network providers can build a scalable infrastructure to meter and analyze their IP metering data, thus they have a more accurate view on the traffic they carry. As a result, service providers can be provided with better quality services, which finally increase the quality of service experience (QoE) for the end-users. By using Linubia, network administrators can have a more granular control and overview on the type and amount of traffic created or received by the users of their network.

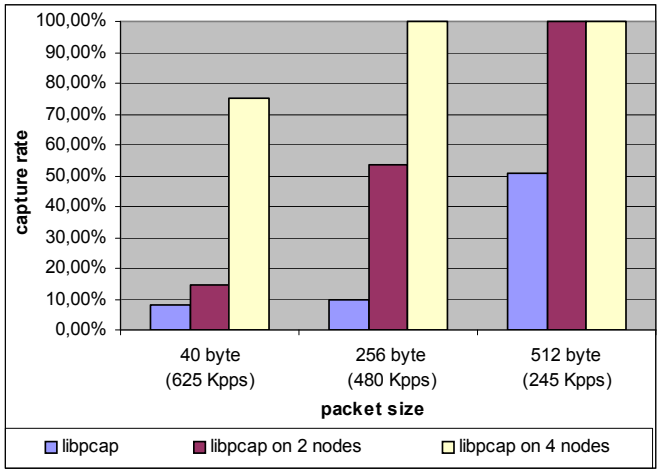

Figure 6: Performance improvement of libpcap

The investigations of this thesis show that by distributing the IP traffic metering data to multiple devices, a scalable infrastructure for IP traffic analysis may be built. Increase in IP traffic can be addressed by adding new devices to the analysis infrastructure. The SCRIPT framework was developed in this thesis to evaluate the new designed mechanisms. SCRIPT distributes flow records to multiple nodes and enables traffic analysis workload to be shared by multiple devices. Traffic analysis applications, like delay measurement or asymmetric route detection, access the SCRIPT functionality over a well-defined API. The SCRIPT framework uses a flexible routing function that can be specified according to the demands of each analysis application separately. It builds on standard protocols and supports IPFIX and NetFlow-based data transfer.

The SCRIPT framework has been implemented as a prototype and evaluated both on standard PC hardware as well as on Cisco AXP cards. The performance evaluations show that SCRIPT increases the total number of flow records processed compared to a centralized solution and it scales with the total number of flow records exported in a network. As its evalua-

tion reveals, the framework distributes flow records nearly equally among all nodes in the SCRIPT overlay, resulting in a fair balance of workload among all nodes.

In case of software-based IP traffic monitoring, at high packet rates the performance bottleneck is the operating system reading the packets from the network interface card. The approach investigated in this thesis is based on (a) cooperation between multiple PCs which receive each a copy of the traffic to be analyzed, and (b) a reduction of the number of packets the operating system needs to copy from the network interface card on every PC. Based on a cooperation protocol each PC can extract from the traffic a subset of packets which it analyzes, such that no packet will be investigated twice. A design and prototypical implementation for such a distributed packet capture mechanism named DiCAP was proposed. DiCAP does not require any dedicated hardware, which makes it a cost-effective solution for capturing IP packet headers at high packet rates using off-the-shelve PCs.

As the link between an end-user (or a process) and an IP packet is lost once a packet leaves a network, the approach taken to account IP traffic on a per-user basis should make use of mechanisms embedded in end-devices, which can intercept network calls of processes an thus map each packet the the process which created (or received) the packet, or the user who owns that process. Linubia, the third mechanism developed in this thesis, shows by its design and prototypical implementation that such a user-based IP accounting approach is technically possible on modern Linux (running kernel version 2.6.x) operating systems. The design is IP protocol independent and can be used for IPv4 as well as for IPv6 traffic in parallel. Linubia's metering module can be easily integrated into an AAA infrastructure. The design presented shows a clear proof of concept which compared to traditional device-based accounting mechanisms allows the mapping of network traffic not only to a device, but more specific, to the user which generated the respective traffic. Performing traffic metering at the linkage point between the networking subsystem and the socket interface allows accessing the process management structures of the operating system. Thus, new interesting mechanisms could be implemented, such as schedulability of processes based on network usage (besides the traditional CPU usage scheme). Linubia could also be used to create new network filters or firewalls that allow for or deny network access to specific applications or users running on a host, instead of only allowing or denying specific services. Additionally, new firewall and traffic scheduling policies could be designed so that a user might be blocked, or his traffic limited, once he exceeds some predefined traffic threshold.

Future work should focus on integrating the current work on IPFIX mediation performed by the IETF in the results of this thesis. The result coulf be the starting point of a standardized distributed IPFIX mediation framework. In the context of the SCRIPT prototype, future work should focus on making SCRIPT more flexible by designing a mechanism that allows SCRIPT applications to be deployed ontop of a running SCRIPT network, without restarting the participant nodes. The current solution requires a recompile and redeployment of the system each time a change is made in an application, or a new application is deployed. A mechanism that allows applications to be added as *plugins* in a running SCRIPT network would significantly reduce both, the time required to deploy a SCRIPT application, and the unavailability of the other SCRIPT application due to service downtime.

ACKNOWLEDGEMENTS

This work was supported in part by the Cisco University Research Program Fund, Grant-No. 2008-02735, the IST Network of Excellence EMANICS funded by the European Union under contract number FP6-2004-IST-026854-NoW, and the DaSAHIT project funded by the Swiss National Science Foundations (No. 200021-118128). The authors would like to thank Ralf Wolter, Benoit Claise, David Hausheer, and Aiko Pras for their discussions and feedback on this work.

REFERENCES

[1] Cisco Systems: *Hyperconnectivity and the Approaching Zettabyte Era*, White Paper, June 2009.

[2] B. Claise: *NetFlow/IPFIX Usage in Network Management*, EMANICS/IRTF-NMRG Workshop, Munchen, Germany, October 2008.

[3] Data Retention Law in Germany: *Gesetz zur Neuregelung der Telekommunikationsüberwachung und anderer verdeckter Ermittlungsmaßnahmen sowie zur Umsetzung der Richtlinie 2006/24/EG*.

[4] Data Retention Law in Italy: Decreto-Legge n.144, 27 luglio 2005.

[5] Endace: http://www.endace.com/, March 2008.

[6] C. Estan, G. Varghese: *New Directions in Traffic Measurement and Accounting*, ACM SIGCOMM Internet Measurement Workshop, San Francisco, California, U.S.A., November 2001, pp. 75-80.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee: *Hypertext Transfer Protocol -- HTTP/1.1,* IETF RFC 2616, June 1999.

[8] C. Morariu, P. Racz, B. Stiller: SCRIPT: A Framework for Scalable Real-time IP Flow Record Analysis. 12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010), IEEE, Osaka, Japan, April 2010.

[9] C. Morariu, B. Stiller: DiCAP: Distributed Packet Capturing Architecture for High-Speed Network Links. 33rd Annual IEEE Conference on Local Computer Networks (LCN), Montreal, Canada, October 2008.

[10] F. Schneider, J. Wallerich: *Performance Evaluation of Packet Cacpturing Systems for High-Speed Networks*, 2005 ACM conference on Emerging network experiment and technology, Toulouse, France, October 2005.

[11] G. Stampfel, W. Gansterer, M. Ilger: Data Retention - The EU Directive 2006/24/EC from a Technological Perspective, Medien und Recht Publishing, 2008.

[12] Wolfgang J., S. Tafvelin: *Analysis of Internet Backbone Traffic and Header Anomalies Observed*, 7th ACM SIGCOMM Conference on Internet Measurement, San Diego, California, U.S.A., October 24-26, 2007.