

CESNET Technical Report 14/2009

Flow Measurement Extension for Application Identification

MARTIN ŽÁDNÍK

Received 11.12.2009

Abstract

Modern networks are expected to provide wide range of application-oriented services. While some applications require a network to be loss-free, low delay with low jitter, others are fault tolerant and happily trade off quality for higher bandwidth. In order to measure these requirements and subsequently provide them, network nodes must be able to determine the application in traffic carried. Since flow measurement is usually utilized to gain information about the traffic mix, we propose to extend it with L7 decoder based on signature matching to identify the part of applications that are not covered by other methods, such as port lookup, fingerprinting and behavioral analysis. As an example, we compare signature matching and port lookup on a CESNET backbone link in order to motivate our future work on a hybrid application identification system based on a combination of several approaches.

Keywords: application identification, flow, measurement

1 Introduction

Accurate network traffic measurements are fundamental for a wide range of network operations such as traffic accounting, traffic engineering, anomaly detection, and long-term forecasting for bandwidth provisioning. Cisco NetFlow or IPFIX [6] are traditional tools to acquire data for these operations but with the rise of peer-to-peer and interactive communication, common means are no longer sufficient to describe the heterogeneity of network utilization by applications. The major shortcoming lies in loosing the ability to identify which applications generate the traffic carried, making it very hard to use gathered flow data for further analysis. This partial blindness compromises operations where flow data serves as a key input for following processing. Moreover, the application identification is expected to be a growing challenge due to several facts:

1. Some applications can be still identified by their assigned transport port number but a growing group utilizes ephemeral port range.
2. Ephemeral ports are used statically as well as dynamically, i.e., part of applications utilize the same port range whereas others choose ports randomly.
3. An application may deliberately masquerade its traffic using ports assigned to other applications (e.g., to evade firewall).

Application identification has drawn much attention and several approaches evolved to address the problem. The most abstract, sometimes called classification in the dark, approach is host behavioral analysis [1], [2]. Its idea is to observe just connections of each host and derive application running on the host by its typical connection signature. If more details of each connection are available, statistical

fingerprinting [3] comes into play. In this case, a feature set is collected per each flow and it is supposed that applications differ in values of the feature set and hence create a unique fingerprint for each application. Behavioral and statistical fingerprinting generally classify traffic to application classes rather than to particular applications. The reason is that different applications doing same task exhibit similar behavior. For instance, application protocols as Oscar (ICQ), MSN, XMPP (Jabber) are all transferring interactive chat communication and hence they have similar behavior, which makes it very hard to differentiate between each other. The inability to distinguish applications within the same class might be seen as a drawback in some situations when, for example, it is necessary to block particular application while allowing other in the same class. On the other hand, classification based on behavioral heuristics is quite robust against evasion and it does not require payload inspection which makes flow measurement a good source of input data. In our case, we can feed previous two heuristics with data collected by hardware Flexible FlowMon probe [8] capable of reporting multiple chosen features per each flow.

The probe is based on a commodity PC running Linux OS with PCI-Express x8 network acceleration card COMBOv2¹. The measurement process is implemented in the acceleration card where the host PC serves as an exporter with capability to perform further aggregation upon reported flows.

In order to support accurate application identification we enhance Flexible FlowMon with an alternative approach addressing application identification. The extension is called L7 decoder and it is based on deep-packet inspection using regular expressions – signature matching. Such a method is prevalent in traffic shaping systems of commercial vendors and if tuned properly it allows to identify application protocols in traffic carried. Moreover, if the result of L7 decoder is combined with flow data already provided by Flexible FlowMon a complete feature base may be created to recover the knowledge of the real application traffic mix.

The report is organized as follows. Motivation and state-of-the-art application identification methods were described first. Second chapter is devoted to hardware description of flow measurement pipeline with L7 decoder and its architecture. Measurement of consumed resources, identification capabilities and performance are described in Evaluation chapter. The paper concludes with a report summary and brief outline of our future work.

2 Hardware Support for Application Identification

Flexible flow measurement inherently supports application identification by reporting flow statistics, but not all of them are relevant and usually some important statistics are missing. Flexible FlowMon already addresses the lack of variability of measurement process by designing a customizable flow record in which items and their corresponding operations may be arbitrarily defined. Therefore it is not an issue to support classification of traffic into abstract classes using behavioral techniques but to extend the ability to distinguish between particular applications is a challenge.

As Flexible FlowMon runs on COMBOv2 board equipped with FPGA, it is

¹ <http://www.liberouter.org/hardware.php?flag=2>

possible to exploit numerous programmable hardware resources to implement L7 decoder with the capability to derive particular application protocol. The core of L7 decoder is based on well studied problem of pattern and regular expression matching which was shown to be suitable for hardware implementation in FPGA [4], [5].

Although payload inspection was probably never meant to be a part of flow measurement, it is straightforward to incorporate result of L7 decoder in the flow record. Each received packet is subjected to payload inspection. Its outcome is attached to a packet and subsequently aggregated into a corresponding item in a flow record.

In order to provide more insight on Flexible FlowMon and its L7 decoder, the whole hardware flow measurement process is described in detail. The flow measurement process is implemented as an application core in NetCOPE framework [7] which serves as a wrapper of FPGA peripherals such as network interfaces and PCI-Express interface and memories (see Figure 1).

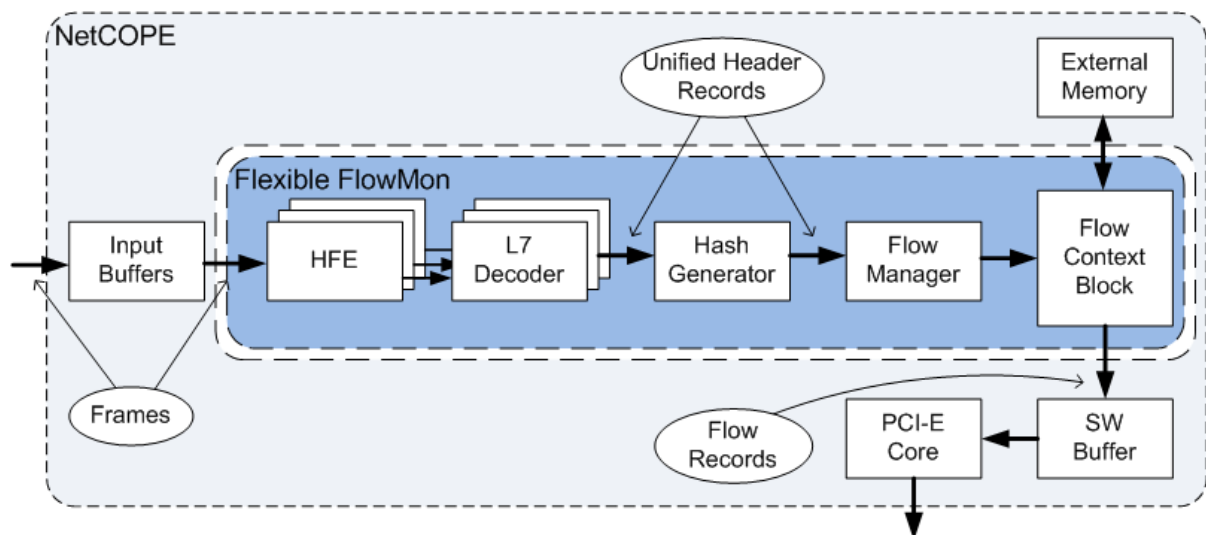


Figure 1. A system architecture of NetCOPE framework with Flexible FlowMon module

The network interface block assigns a timestamp to each correctly received frame and forwards it to the processing pipeline. At the beginning of the flow measurement pipeline, the input frame is parsed layer by layer in Header Field Extractor (HFE), relevant fields are extracted and grouped into so-called Unified Header Record (UHR). Reaching the end of the protocol stack identifies the start of application payload, which is scanned for known application signatures in L7 decoder. These signatures are described in the form of PCRE² (Perl Compatible Regular Expression) signatures, for example:

```
# SMTP - Simple Mail Transfer Protocol
/^220[\x09-\x0d -~]* (e?smtp|simple mail)/i
```

Each PCRE gets transformed through a sequence of phases into a minimized deterministic finite automaton (DFA), as depicted in Figure 2.

² <http://www.pcre.org>

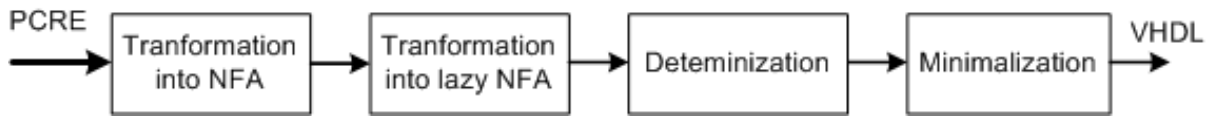


Figure 2. Transformation sequence of PCRE into deterministic finite automaton described in VHDL

The first stage is a transformation of PCRE into nondeterministic finite automaton (NFA) representation. Subsequent transformation to lazy NFA allows to increase the throughput by accepting multiple input characters in a single transition. Finally, the lazy NFA is made deterministic and the number of its states reduced so that it could be compiled into VHDL. The VHDL implementation is further optimized by adding shared character decoder translating each input character or a class of characters to one-hot encoding. Therefore, the decision whether to perform a transition is based on an assertion of a single bit thus making the next-state logic really simple. Further, a symbol table must be constructed to support processing of multiple characters in a single transition by describing all acceptable character sequences. The architecture of L7 decoder is displayed in Figure 3.

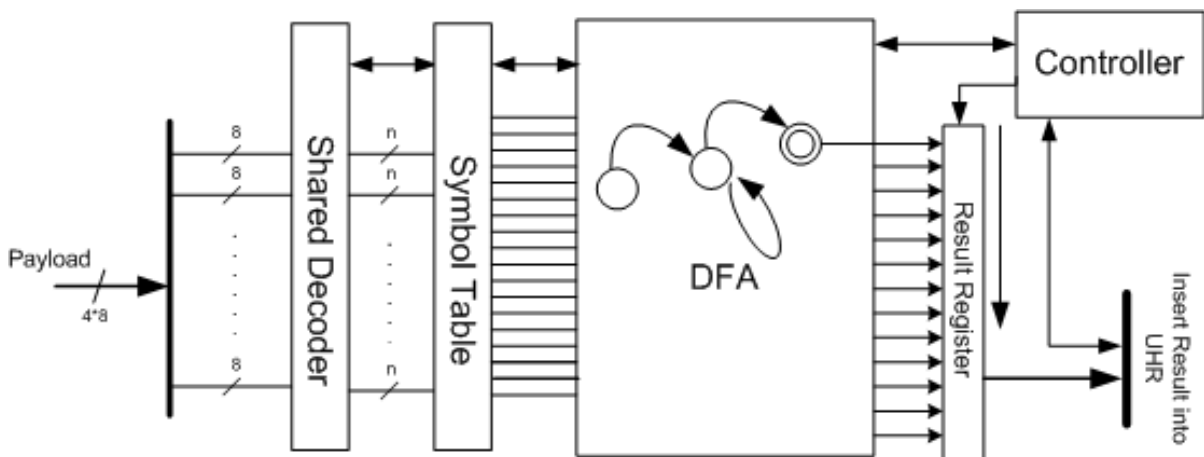


Figure 3. Inner structure of L7 decoder based on regular expression matching.

Its input is created by a frame with the position of the start of payload data and UHR. Based on the position, the L7 decoder starts to scan the payload for all synthesized signatures in parallel. The result is encoded as a bitmap, one bit per each signature. The bitmap is inserted into an allocated space in the UHR. At the end of the flow measurement pipeline is Flow Context Block, where the UHR gets aggregated into a flow record, an OR function (or any other) may be used to mix previous match results with the actual one, hence accumulating all signatures that were matched in the given flow.

Both extraction and payload inspection are computationally intensive tasks and must be instantiated in multiple branches to cope with incoming packet rate as displayed in Figure 1. Additional logic provides support of correct ordering of arriving frames and their corresponding results represented by UHR. A reorder may happen when subsequent frame is processed by a branch that is less occupied or the processing takes shorter time, for example, due to shorter length of a packet. A

solution that was implemented is to assign a ticket to each frame, then propagate it into the result and finally correctly reorder UHR using these tickets.

The rest of the flow measurement pipeline consists of Hash Generator, Flow Manager and Flow Context Block. Hash Generator computes Bob Jenkins’s hash upon key fields stored in UHR, i.e., upon fields that determine which flow the packet belongs to. The computed hash serves as an index into a flow table as well as a fingerprint to identify distinct flows. By accurately dimensioning the length of a hash, one can minimize collisions to an acceptable rate. Flow Manager keeps state of each flow from a temporal point of view to identify flows that are no longer alive or live too long so that these may be reported. At the end of a pipeline every UHR is aggregated into corresponding flow record by Flow Context Block.

3 Experimental Evaluation

The first set of experiments was focused on finding out the relationship among the number of signatures, theoretical throughput and resource consumption. Better understanding of this relationship is important to correctly estimate the number and structure of signatures that fit into FPGA. The amount of occupied resources by L7 decoder depends on the throughput (number of processed bytes per clock cycle * frequency) and on the number of signatures synthesized; these dependencies are plotted in Figures 4 and 5, respectively.

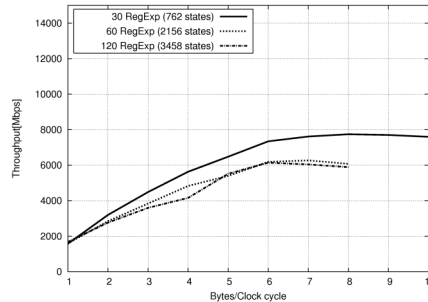


Figure 4. Throughput of L7 decoder at maximum synthesizable frequency (randomly generated patterns)

Based on our resource utilization experiments, we used three instances of L7 decoder-HFE pairs. Each L7 decoder is capable of processing four bytes per clock cycle and all together provide theoretical throughput of 9.4 Gbps at 100 MHz frequency. Such throughput is sufficient to process 10 Gbps traffic because only payload of a packet is processed while its header is skipped. Overall consumed resources on Virtex-5 LXT155 are denoted in Table 1.

Table 1. Virtex-5 LXT155 resource utilization of design variants.

Design variants	Slices	BlockRAMs
Flexible FlowMon	54%	27%
Flexible FlowMon + L7	71%	61%

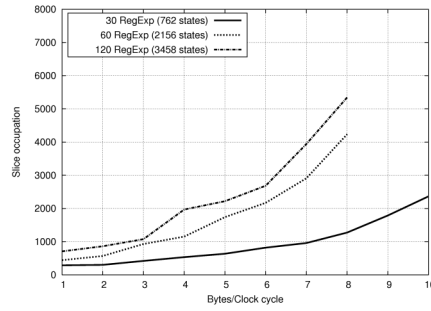


Figure 5. FPGA resource utilization by L7 Decoder (randomly generated signatures)

Functionality and performance were subjected to the second set of experiments. In order to evaluate functionality of L7 decoder, the rear part of the design of Flexible FlowMon was removed. The interest was to verify that each arriving packet sets the corresponding bit in the bitmap if the packet contains a signature. The flow measurement pipeline contained only HFE, L7 decoder and Hash Generator so the software part of the probe observed all Unified Header Records rather than aggregated flow records. Using this setup, the design was synthesized with several artificial signatures. When tested under the generated traffic it found all signatures that were contained in the packet. Moreover, 32 signatures of L7-filter³ project were synthesized afterward to demonstrate one of possible use cases. As an example, the L7 decoder is used to reveal percentage of application traffic that is transferred on other ports than those assigned to it and also to reveal how much traffic cannot be recognized as a valid application traffic on a well-known port. The experiment lasted about half an hour and was performed on a 10 Gigabit CESNET backbone link between Brno and Prague. Both directions of the link were monitored via a SPAN port of Cisco router. The actual link utilization was 5.6 Gbits/s and 683 Kpkts/s.

Observed UH records were aggregated into flow records. The application field is aggregated by an OR function over all UHRs of the flow. Not surprisingly, large part of flows did not match any signature due to one of five reasons:

1. SPAN port may loose packets or trim payload if under heavy load,
2. traffic belongs to an application which is out of the scope of our 32 application signatures,
3. the flow started prior to measurement and only the end of flow was observed where no signature matched,
4. the flow did not proceed that far to exhibit a known signature, e.g., did not finish TCP handshake,
5. the signature is not strong enough to cover application of interest in all its variations.

On the other hand, the number of identified packets and identified bytes is much higher. If one or more signatures were found in a flow then all its packets

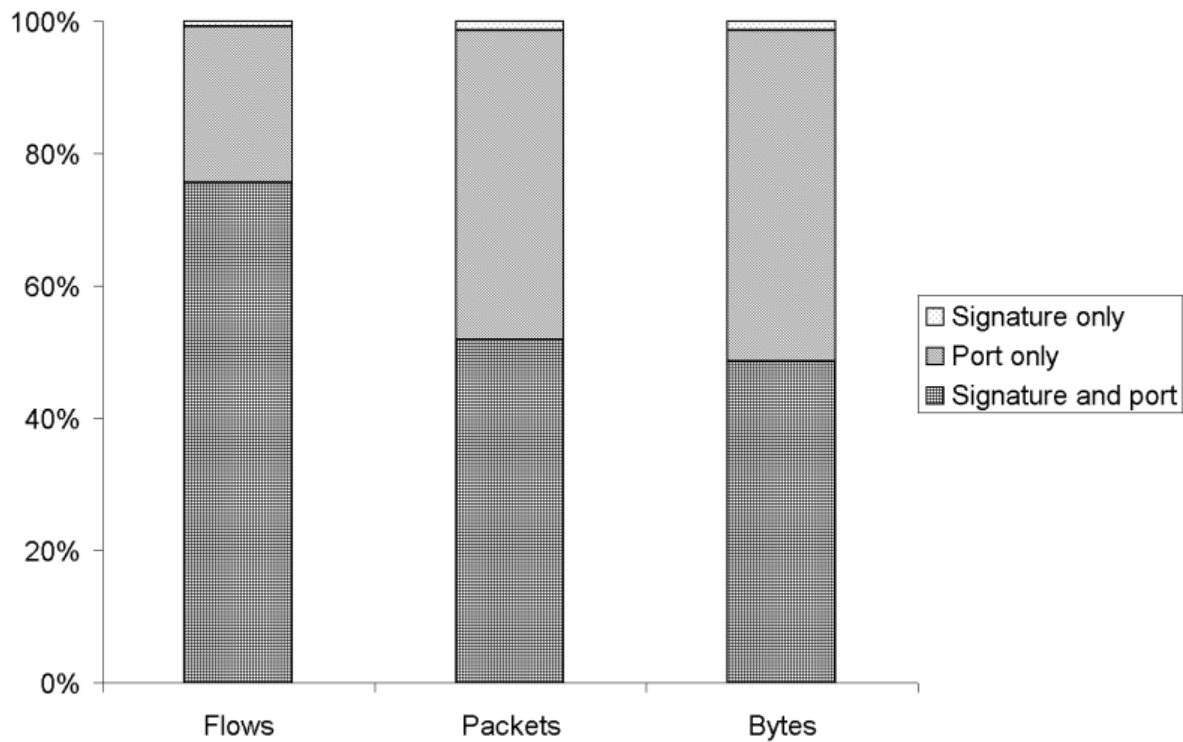
³ <http://l7-filter.sourceforge.net/>

Table 2. Comparison of identified and unidentified flows, packets and bytes by L7 decoder.

	Flows	Packets	Bytes
All	48 M	1.6 G	1.3 TB
Identified	43%	74%	81%
Unidentified	57%	26%	19%

and bytes were denoted as identified. Table 2 presents comparison of identified flows, packets and bytes.

Since this was an on-line measurement, there was no annotated data set, i.e., a network traffic file where each packet was assigned an application class based on supervised/manual analysis, therefore it was not possible to distinguish among those five reasons of identification failure, hence presented results might be strongly influenced. Figures 6, 7, 8 and 9 demonstrate results of HTTP, RTSP, SIP, BitTorrent and eDonkey identification and compare signature matching with port based approach. Each graph displays a flow/packet/byte distribution in particular application protocol category. Clearly, both approaches have their own benefits and drawbacks which are discussed in the light of each application identification.

**Figure 6.** HTTP protocol identification

Identification of HTTP protocol based on both signature and port number is quite successful, both methods gave the same result that about 80% of flows, accounting for approximately 50% of packets and bytes, is indeed an HTTP traffic. About 20% percent of flows, solely identified by port number 80, accounts for about 50% of packets and bytes. Such an imbalance is rather suspicious and may reflect that a foreign traffic is tunneled through port 80. Very small part of traffic was

identified as HTTP solely based on a signature. Most of these flows targeted HTTP proxies on port range 8000 to 9000.

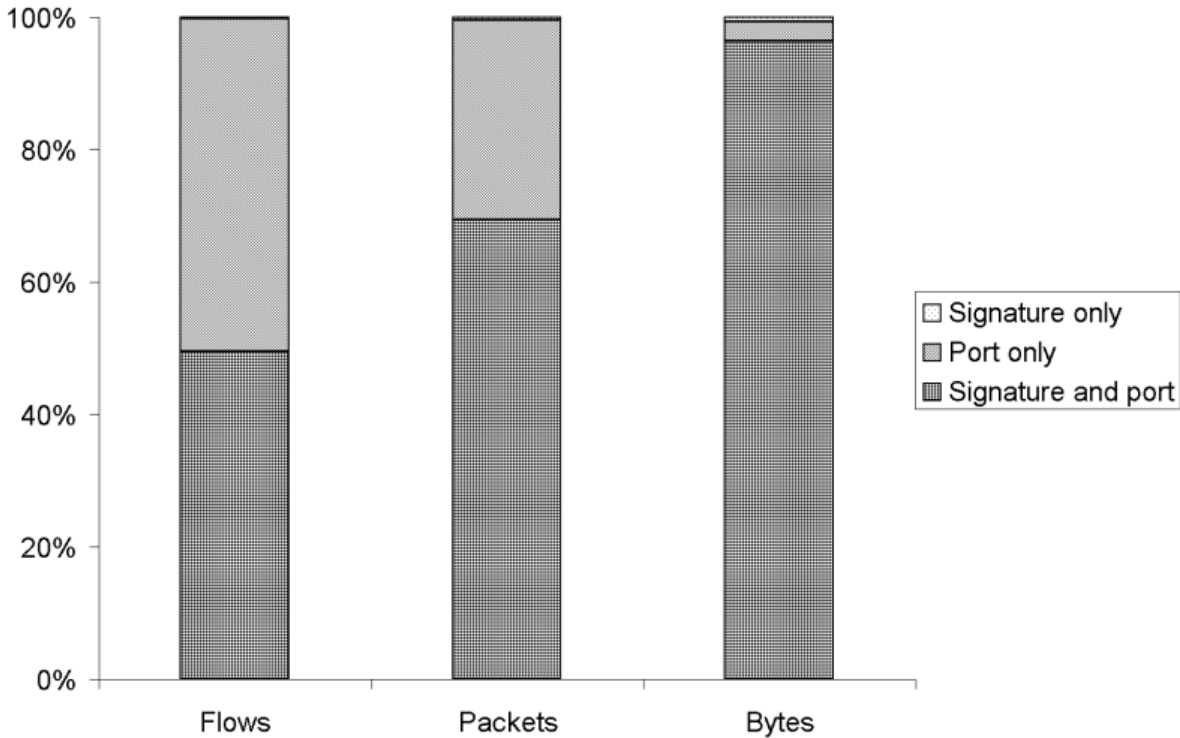


Figure 7. RTSP protocol identification

In case of RTSP, signature matching identifies only half of flows that are identified by port number 554 but at the same time both methods yielded that 97% of bytes were RTSP traffic. This behavior is most likely caused by the definition of a signature that matches RTSP connections containing only SUCCESS status code.

Peer-to-peer protocols such as BitTorrent and eDonkey are better identified by L7 decoder, see Figure 8 and 9. The reason is that the official port numbers for BitTorrent are in the range from 6880 to 7000 but clearly users can change it (and they often do). On the contrary it is expected that peer-to-peer applications communicate using dynamic port range therefore their identification based on port numbers is pointless.

Performance measurement was carried out to reveal throughput of the trimmed design. The dependency between packet length and throughput is plotted in Figure 10. The pipeline is composed of three parallel instances of L7 decoder-HFE pairs to reach the wire speed throughput. The graph shows that certain lengths cause a minor drop of performance. We analyzed this issue in depth and discovered inappropriate distribution policy being used in the frame splitter in front of HFEs.

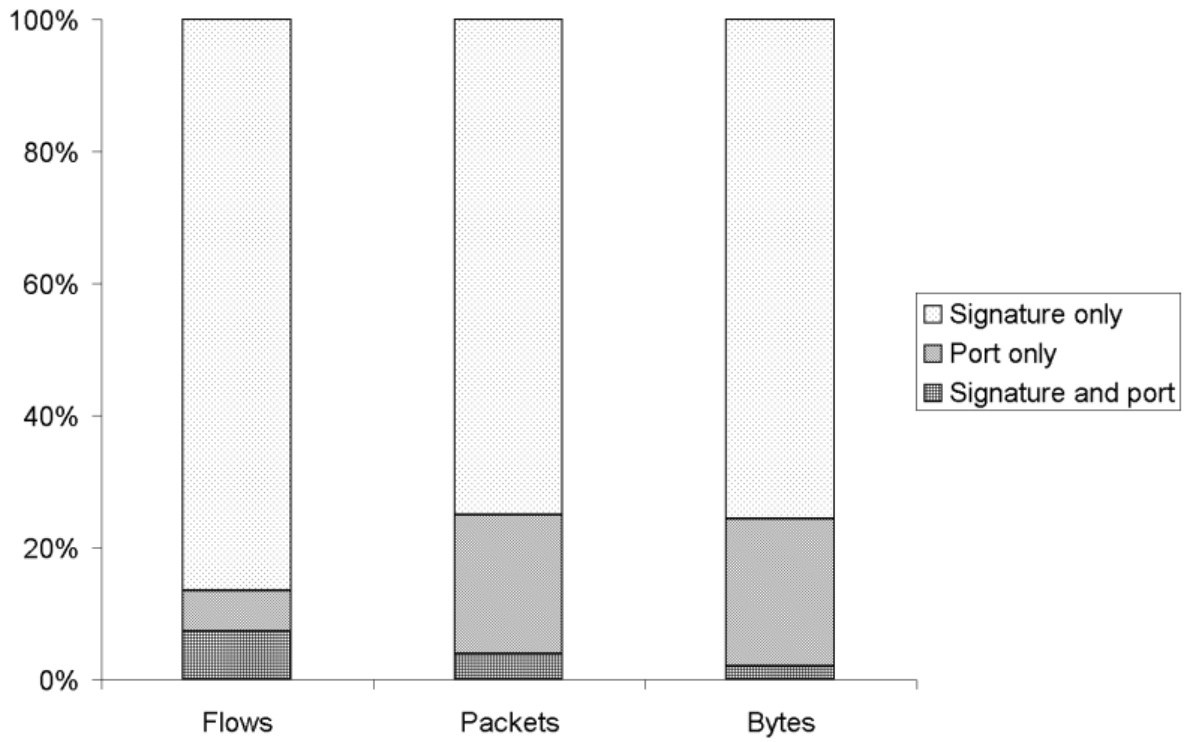


Figure 8. BitTorrent protocol identification

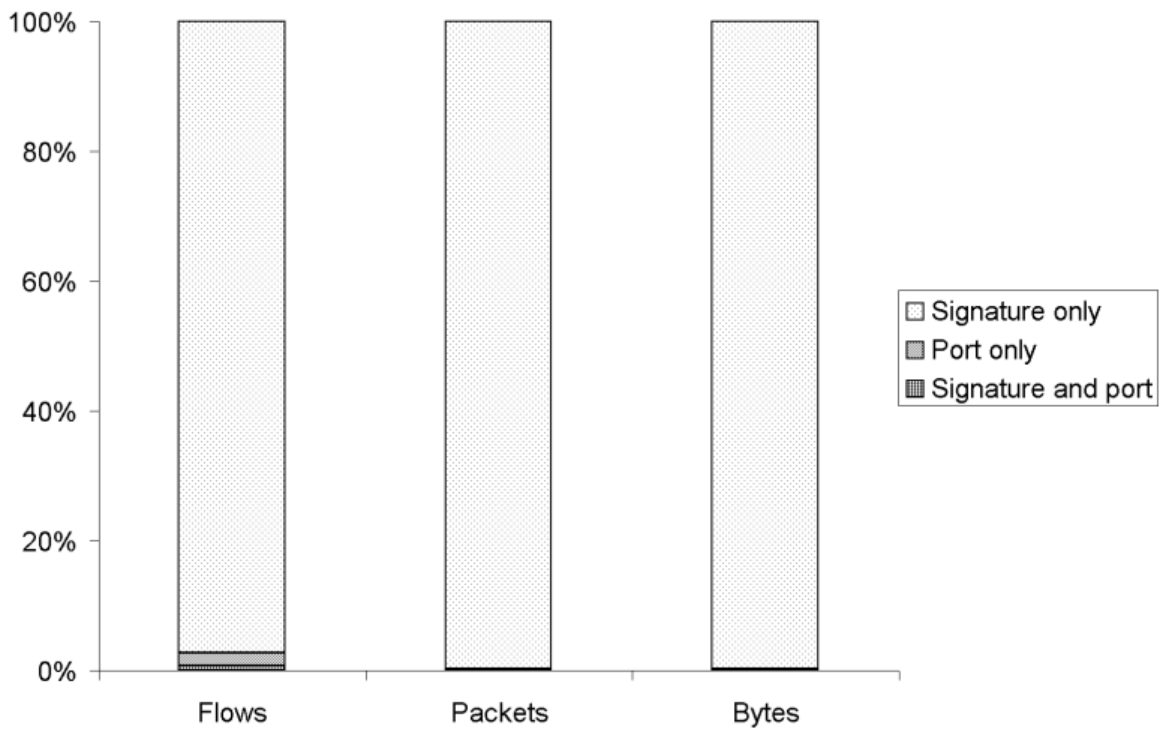


Figure 9. eDonkey protocol identification

4 Conclusion

Flow measurement is a popular way of acquiring a knowledge about network traffic

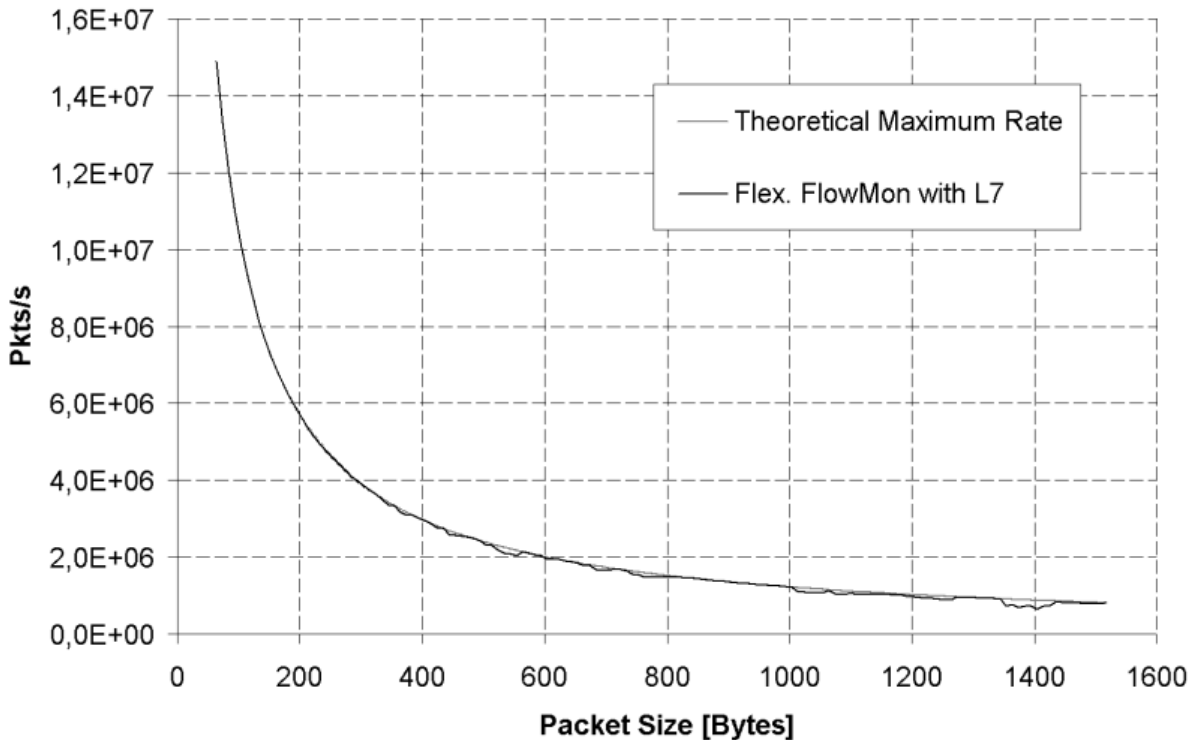


Figure 10. Performance of Flexible FlowMon with L7 extension

and the network itself. Administrators typically use flow measurement to observe network utilization, reveal significant sources of data transfers, debug incidents and detect anomalies. To support these fundamental network operations, flow measurement has evolved to its current standard in the form of IPFIX RFC [6]. We follow this evolution and extended capability of flow measurement element, namely Flexible FlowMon probe, even further with application identification module – L7 decoder – based on signature matching. The task of L7 decoder is to provide information about application being carried in the packet payload which would be dropped in a standard flow measurement process.

The L7 decoder is based on matching application signatures, represented as regular expressions, in the packet payload. Each signature is translated and optimized in several stages into deterministic finite automaton. Final DFAs are described in VHDL and synthesized with the flow measurement process into a bit-stream loadable into FPGA. The advantage of FPGA signature matching is the execution of all automata in parallel, hence achieving high throughput. The signature matching is performed on a per packet basis and its outcome is a bitmap which is aggregated into a flow record.

The experiments made explore FPGA resource utilization when L7 decoder is added into a flow measurement pipeline. It was shown that a payload inspection at ten gigabits using 32 standard application signatures can be achieved at the cost of approx 1/4 of Virtex-5 LXT155 resources. Practical tests were carried out on a real network to present benefits and drawbacks of application identification based on signature matching in comparison to port approach. The results show that L7 decoder may add valuable information to a standard flow record if used and inter-

preted properly.

The report concludes with performance measurement of Flexible FlowMon extended with L7 decoder based on 32 signatures adopted from L7-filter project. Measured performance proves that with minor deviations the ten gigabit throughput was reached.

Our future work is to improve the quality of L7 decoder by tuning application signatures as well as to extend functionality of L7 decoder for stateful protocol analysis rather than simple regular expression matching. We also envision and intend to build a high-performance application identification system based on Flexible FlowMon which will combine several methods, such as port-, behavioral- and signature-based, to achieve more accurate results.

4.1 Acknowledgement

This work is supported by the research intent MSM6383917201.

References

- [1] KARAGIANNIS, T.; PAPAGIANNAKI, K.; FALOUTSOS, M. BLINC Multilevel Traffic Classification in the Dark, In *ACM SIGCOMM 2005*, Aug 2005.
- [2] JOHN, W.; TAFVELIN S. Heuristics to Classify Internet Backbone Traffic based on Connection Patterns. In *International Conference on Information Networking (ICIN)*, Sept 2008.
- [3] MOORE, A. W.; ZUEV, D. Internet traffic classification using bayesian analysis techniques. In *SIGMETRICS 05: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Banff, Alberta, Canada, June 2005, p. 50-60.
- [4] SIDHU, R.; PRASANNA, V. K. Fast Regular Expression Matching Using FPGAs. In *Proceedings of the the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Washington, DC, 2001.
- [5] SOURDIS, I.; PNEVMATIKATOS, D. N. Fast, Large-Scale String Match for a 10Gbps FPGA-Based Network Intrusion Detection System. In *Field Programmable Logic and Application*, 13th International Conference, Lisbon, Portugal, 2003, p. 880-889.
- [6] CLAISE, B. (ed.). *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information*. RFC 5101⁴, IETF, Jan 2008.
- [7] MARTÍNEK, T.; KOŠEK M. NetCOPE: Platform for Rapid Development of Network Applications. In *Proc. of 2008 IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop*, Bratislava, 2008.

⁴ <http://tools.ietf.org/html/rfc5101>

- [8] ŽÁDNÍK, M.; ŠPRINGL, P.; ČELEDA P. *Flexible FlowMon*. Technical report 36/2007⁵, Praha: CESNET, 2007.

⁵ <http://www.cesnet.cz/doc/techzpravy/2007/flexible-flowmon/>