Rodríguez de los Santos, G., Hernández, J. A., Urueña, M. & Muñoz, A. (2014). A Bloom Filter-Based Monitoring Station for a Lawful Interception Platform. In: *Multimedia Communications, Services and Security (MCSS 2014). Communications in Computer and Information Science (CCIS)*, 429, pp. 214-228. Switzerland: Springer.

DOI: 10.1007/978-3-319-07569-3_18

A Bloom Filter-based monitoring station for a Lawful Interception Platform

Gerson Rodríguez de los Santos, Jose Alberto Hernández, Manuel Urueña, and Alfonso Muñoz[1]

Universidad Carlos III de Madrid
Avda Universidad 30, 28911 Leganés, Madrid, Spain
[gsantos, jahgutie, muruenya, ammunoz]@it.uc3m.es,
WWW home page: http://www.it.uc3m.es/

Abstract. Lawful Interception (LI) is a fundamental tool in today's Police investigations.Therefore, it is important to make it as quickly and securely as possible as well as a reasonable cost per suspect. This makes traffic capture in aggregation links quite attractive, although this implies high wirespeeds which require the use of specific hardware-based architectures. This paper proposes a novel Bloom Filter-based monitoring station architecture for efficient packet capture in aggregation links. With said Bloom filter, we filter out most of the packets in the link and capture only those belonging to lawful interception wiretaps. Next, we present an FPGA-based implementation of said architecture and obtain the maximum capture rate achievable by injecting traffic through four parallel Gigabit Ethernet lines. Finally, we identify the limitations of our current design and suggest the possibility of further extending it to higher wirespeeds.

Keywords: Lawful Interception, FPGA, Bloom filter, Packet Capture

1 Introduction

Criminality is, and is likely to be at all times, a great problem in our society. The first task in solving a criminal case concerns the collection of evidence and the investigation of suspects. In some occasions, the Police forces need to lawfully intercept the communications (phone, computers, etc) of suspects, especially in severe crimes like terrorism, child pornography, political corruption and organised crime in general. On the other hand, the secrecy of communications is recognised as a fundamental right in most countries, therefore a wiretap warrant, issued by a judge, is necessary to intercept communications.

A usual approach for Lawful Interception (LI) involves wiretapping directly the suspect's subscriber loop, typically cable (xDSL, HFC), fibre-based (FTTx) or wireless (WiFi). However, individual wiretapping poses serious scalability problems, especially concerning cost. Besides, wiretapping at aggregation points, where the traffic of thousands of users is aggregated, is technologically challenging, but possibly more cost-effective. Such a large-scale monitorisation requires

fast traffic capturing (at the line rate), and high-speed filtering out of non-suspicious traffic while ensuring that 100% of the suspects' traffic is captured. To accomplish these goals, specialised monitoring hardware is required, for instance FPGA- or GPU-based systems.

Because high processing speeds are required in these environments, the use of data structures that provide low latency is required as well as specialised hardware. One possible solution which has been the subject of study in recent years is the use of Bloom Filters [1] (BF). These data structures permit to quickly check if a binary string belongs to a registered set of elements with a reasonable false positive probability.

In this light, this paper proposes an FPGA-based monitoring station architecture for the high-speed collection of suspects' traffic in multi-Gbit/s links. Additionally, we present an implementation based on a NetFPGA with 4x 1 Gbit/s input ports where traffic is captured, filtered and forwarded to other output port(s). Inside the monitoring station, the source and destination IP addresses of every incoming IP packet are checked against a list of suspect's IP addresses which are stored in a Bloom Filter and implemented in hardware. This implementation allows the monitoring station to operate at wirespeed, showing its applicability in realistic investigation scenarios.

The rest of the paper is organised as follows: Section 2 reviews previous work related with high-speed packet processing, GPUs, multicore processors and FPGA based systems. Section 3 overviews the fundamentals of Bloom Filters. The whole Lawful Interception scenario and the inner hardware architecture are described in Section 4. Finally, Section 5 shows introduces the FPGA platform used in the implementation of our prototype, as well as a number of experiments demonstrating the feasibility and performance of our prototype. Section 6 concludes this paper with a summary of its main contributions.

2 Related work

While guaranteeing real-time processing and null network information loss is a tall order, there are several technologies appointed for this demanding task, namely GPUs, Multicore processors and FPGAs.

2.1 GPUs

GPUs provide a massive amount of small computational elements, which are very suitable for tasks that can be parellelised at a reduced cost.

GPUs have been used to provide parallelisation to a wide variety of networking tasks. In [2], a GPU-based routing implementation with Deep Packet Inspection (DPI) capabilities is presented. Such a DPI is both implemented using a Finite State Automata (FSA) and a Bloom Filter paradigm, and are subsequently compared showing that Bloom Filters provide the best performance.

GPUs in the routing context were first presented in [2] and further extended in [3]. In this work, the authors show a direct table lookup (with up to 2^{24}

entries) for routing which highly minimises memory access in comparison with other data structures such as Tries.

In [4], an architecture for packet signature matching is examined and implemented in a GPU. A comparison between Deterministic Finite Automata (DFA) and eXtended Finite Automata (XFA) based architectures is provided. Both of them are analysed and implemented for comparison showing a better performance and less memory usage coming from XFA implementations.

In [5], another GPU-based packet regular expresion matching engine is introduced. With three optimisation techniques aimed at improving memory access, the implementation is able to reach 128.6*Gbps* rate.

GPUs have also been a subject of research in the design of Intrusion Detection Systems (IDS). In [6], a GPU parallelised architecture based on the Wu-Manber algorithm is shown. The work in [6] was subsequently improved in [7], where a hierarchical parallel machine architecture on GPU was used to address some of the shortcomings revealed in [6], mainly the problem of state explosion that appears when it is necessary to search for complex regular expressions.

2.2 Multicore processors

Another means of parallelisation comes from multipurpose, multicore processors. In this field, extensive research efforts have been conducted in the parallelisation of packet processing, especially pattern matching, see [8–10]. As a consequence, DPI has been the most recurring topic in this area. In [11] a parallelisation of the L7 filter [12], a DPI extension for Linux Netfilter is presented. Finally, [13] proposes a pre-filtering algorithm to ignore unwanted matches for L7 filter. This allows the L7 filter to get a better efficiency for the L7 rule matching algorithm.

2.3 FPGAs/ASICs

FPGAs and ASICs allow full task customisation and implementation in hardware.

FPGAs have been used to overcome the limitation of pure software environments for traditional networking tasks such as IP forwarding [14]. The most popular application of FPGAs in the networking area comprises those related with pattern and string matching for IDS and Intrusion Protection Systems (IPS). In [15–18] a parallel Bloom Filter based architecture is presented and subsequently improved. Another approach for IDS implementation in FPGA is the Finite State Machine (FSM) paradigm [19, 20], which has different pros and cons with the Bloom Filter approach [21]. Additionally, in [22], an FPGA implementation of a Deep Packet Inspection architecture with Regular Expression Detection is shown. In [23], a parallel pattern matching architecture based on a compact reconfigurable filter and a coprocessor for FPGA is presented.

Finally, there has also been research regarding hardware implementations of firewalls. In [24], [25] two different firewall implementations are shown. There are also some combined proposals like [26], which suggest a combination of Firewall, IDS and rate limiting in the same implementation.

The main contribution of our paper consists in the design of a hardware Bloom Filter packet monitoring station architecture for Lawful Interception.

3 Bloom Filters background

A Bloom Filter [1] (BF) is a data structure used to test whether an element belongs to a certain set or not. A BF is characterised by a number k of hash functions and a binary array N of bits initially set to zero, as shown in Fig. 1.

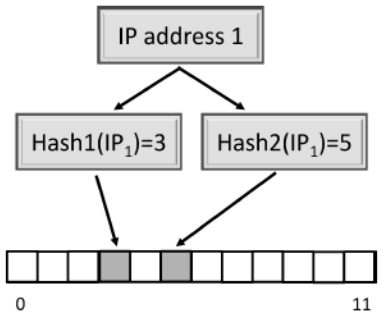


Fig. 1. Bloom Filter example.

Now, consider we wish to store n elements in the array, say for instance the list of IP addresses from suspects to be monitored. In this light, the k hash functions are applied to each element in the IP list producing a number of positions in the binary array, which are then set to one. In the example, $k = 2$ hash functions are applied to the first IP address, setting the third and fifth position of the binary array to unity (note that the binary array has 12 positions, starting from 0 until 11). After the n IP addresses are stored, the binary array contains a number of ones which characterises the list of n IP addresses to be monitored. This is often referred to as the *training phase* of the Bloom Filter.

The average number of ones in the binary array is:

$$E(W) = N\left[1 - \left(1 - \frac{1}{N}\right)^{kn}\right] \approx N\left(1 - e^{-\frac{kn}{N}}\right) \tag{1}$$

where $E(W)$ is often referred to as the *weight* of the BF.

This structure allows to fast check whether an IP address belongs to the set of suspicious IPs stored in the array, just by computing the k hashes and checking the associated positions.

However, the BF may produce false positives, that is, a certain IP address may not have been programmed in the binary array, but still the hash functions applied to it may point at positions set to one. This occurs with the following

probability [1]:

$$P_{fp} = \left(\frac{E(W)}{N}\right)^k \approx \left(1 - e^{-\frac{kn}{N}}\right)^k \tag{2}$$

The false-positive probability reduces for large values of N. Nevertheless, filtering is required to remove the false positives from the actual positives. Given the fact that a Lawful Interception platform must not store traffic that does not belong to the wiretapped suspects, this issue has been addressed in our design and is further explained in section 4.1.

4 System design and architeture

4.1 Lawful Interception Platform architecture

Consider the lawful interception scenario of Fig. 2, further explained in in [27], where the Internet connections of multiple (typically thousands) *subscribers* are aggregated at the Metropolitan Area Network (MAN). We assume that some (very few) of these subscribers are criminal *Suspects* under investigation. To investigate these suspects, a Digital Wiretap Warrant (DWW) [28] issued by a judge is mandatory.

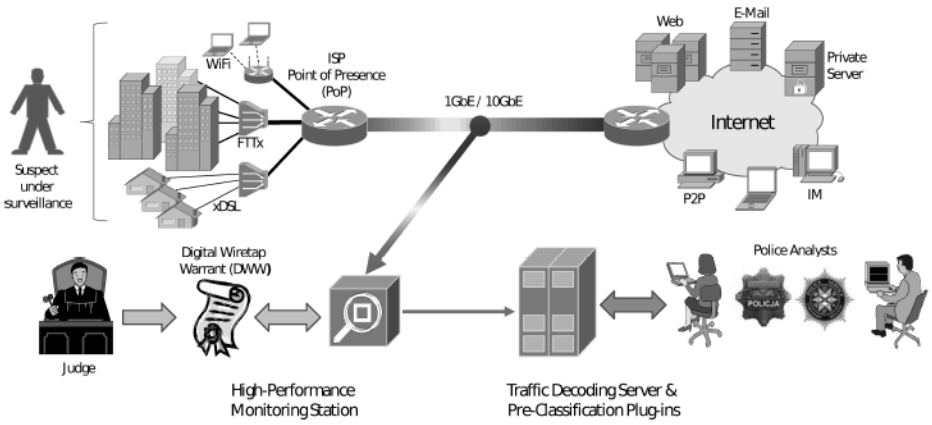


Fig. 2. Lawful Interception scenario

In this scenario, our *monitoring station* collects the traffic of thousands of users traversing its ports. A DWW is required for any capturing process. This means that the monitoring station will not capture any traffic from that suspect unless a valid DWW is provided in order to prevent misuse by the Police, the ISP or any other unauthorised third party. The monitoring station has a list of IP addresses (of the suspects under a wiretap warrant) loaded in its internal Bloom Filter to decide which packets are to be captured and stored for further

investigation by the Police forces. Because Bloom Filters have false positives, packets are filtered at software level at the monitoring station. This ensures that any packet which does not belong to a suspect is never stored. Additionally, zero-loss packet capturing is mandatory for the suspects' traffic.

The captured traffic is then sent to the *Traffic Decoding Server* [27], which takes the capture files produced by the Monitoring Station to reconstruct the files and contents that the suspect has transmitted or received.

4.2 Traffic inspector module architecture

Fig. 3 shows the architecture of the Bloom Filter traffic inspection module, which comprises:

- Two FIFO queues to regulate input/output to the module, called Input and Output FIFO respectively.
- A Packet Buffer in which some packet words are copied for inspection.
- An Inspector module which checks both source and destination IP addresses to be inspected.
- User-Space Bloom Filter Interaction module (USBI), which communicates the User Space software and the Bloom Filter. This module allows to add/remove/update IP addresses in the Bloom Filter dynamically.
- A Bloom Filter, which is queried by the inspector module twice per packet. The Bloom Filter is implemented using the FPGA's BRAM (Block Random Access Memory). The read and write access to the Bloom Filter is managed by a priority-encoded controller, which gives priority to the Inspector over the User Space Bloom Filter Interaction module.

To illustrate how our hardware architecture operates, consider the arrival of a packet at the Input FIFO (step 1 in Fig. 3). The first bytes of the packet are stored in a Packet Buffer (step 2), and simultaneously they are copied to the Output FIFO. Next, the inspector obtains the bytes that contain the source and destination IP addresses of the packet (step 3), and checks whether or not there is a positive matching in the Bloom Filter (step 4). If there is a positive match, then the packet is captured by the monitoring station by changing certain bits in the control header of the packet (step 5). Otherwise, the packet is simply not stored.

Once the first word of the packet is allowed to exit the module, then its next words are automatically forwarded without further checks. After the first word of this packet exit the Inspector module, the next packet in the queue can be analysed.

The priority encoding of the Bloom Filter guarantees that any operation from user space which involves the Bloom Filter (reading or writing) will be delayed if the inspector needs to check the Bloom filter to classify a packet. A simple request-response protocol is implemented between the USBI and the Bloom Filter and also between the USBI and the user space to indicate when low-priority operations have been attended by the Bloom Filter. Software tools

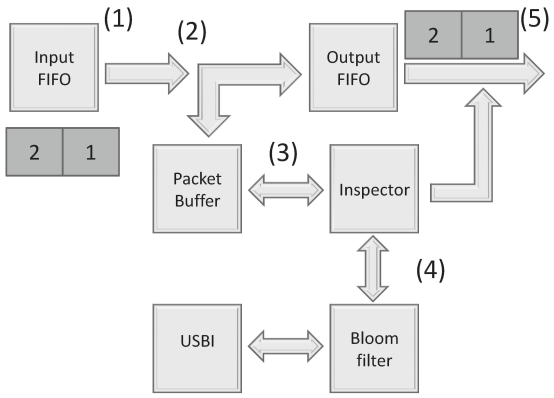


Fig. 3. Block architecture of the traffic inspection module

are used at user space to ensure that the Bloom Filter is properly recalculated each time a new IP address is added or deleted from the list.

Two different interception modes are available in this prototype, namely *Forward and Tap* and *Tap and Drop*. In *Forward and Tap* mode, if a packet matches the search criteria, the packet is forwarded through an output port and a copy of it is also sent to user space for capture. If not, the traffic is simply forwarded transparently to the output port. The *Forward and Tap* mode is intended for a case in which we need to capture in serial mode. This means that the traffic inspector is placed in the middle of the network wire.

In the *Tap and Drop* mode, if a packet matches the search criteria, it is forwarded to user space. If not, it is simply discarded. The *Tap and Drop* mode is the mode that should be used if we want to have a parallel communication interception. In this mode, a copy of the traffic which is being forwarded through the network is sent to the monitoring station. This mode allows to tap more lines than the forward and tap mode because not as many ports are needed to forward the traffic, so the packet inspector has simply to decide if the traffic is to be processed or discarded.

4.3 Design of the Inspector Bloom Filter

This section evaluates the performance of the current implementation of the Inspector Bloom Filter in the NetFPGA. Taking information from a Spanish ISP, we assume that an aggregation metro node concentrates the traffic of about 40000 DSL subscribers producing an average bit rate of 120 Kbit/s each. With these numbers, the average bitrate traversing the NetFPGA:

$$40000 \times \ 120 \text{ Kbit/s} = 4.8 \text{ Gbit/s}$$

which is close to the maximum input bitrate that the 1G NetFPGA may handle. It should be noted that, although the NetFPGA used for our implementation

does not fully support such an input bitrate, our implementation is intended for the validation of our design.

On the other hand, the Bloom Filter implementation comprises $k = 2$ Fibonacci hash functions [29] and a bit array of $N = 65536$ bits. The first hash function h_1 operates directly on the IP address (either source or destination) whereas the second one h_2 performs the hash of a fixed permutation of the bits comprising the IP address. These two hash functions determine which bits have to be set in the Bloom Filter.

Fig. 4 gives the false-positive probability for a number n of IP addresses to be loaded on the Bloom Filter. As shown, for $n = 1000$ suspect IP addresses, the false positive probability equals $9.03 \cdot 10^{-4}$. This value reduces to $9.28 \cdot 10^{-6}$ for $n = 100$ addresses, and to $9.31 \cdot 10^{-8}$ for $n = 10$ IP addresses.

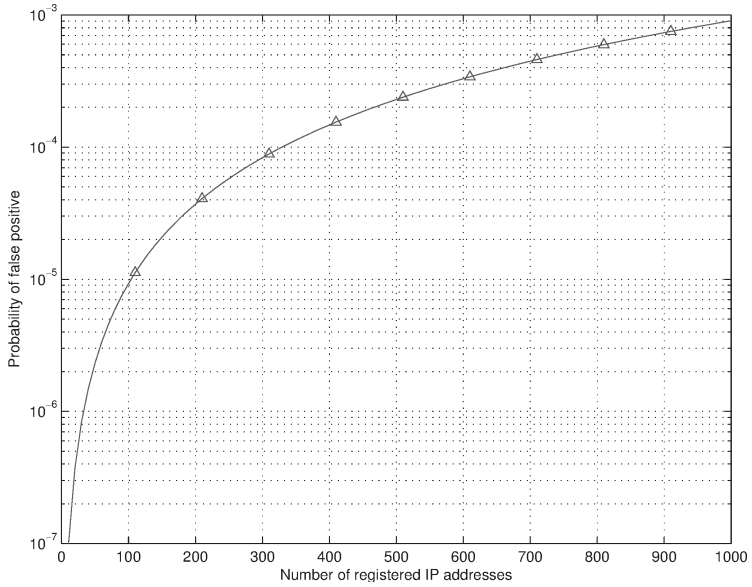


Fig. 4. Probability of false positive in the designed Bloom Filter for k=2, n=65536 bits

These numbers are somehow reasonable since we do not expect more than 1000 suspects in a population of 40000 users. Actually, it is more reasonable to expect 10 to 100 suspects in such a population than 1000 suspects.

Finally, the total incoming traffic arrival at the PCI-X concerns both the actual suspicious traffic plus that traffic due to false positives, i.e.:

$$120 \text{ Kbit/s} \times (n + 40000 \cdot P_{fp}(n))$$

This gives a total data rate of:

$$n = 1000 \text{ Bitrate} = 124 \text{ Mbit/s}$$
$$n = 100 \text{ Bitrate} = 12 \text{ Mbit/s}$$
$$n = 10 \text{ Bitrate} = 1.2 \text{ Mbit/s}$$

In general, we observe that:

$$120 \text{ Kbit/s} \times (n + 40000 \cdot P_{fp}(n)) \approx 120n\text{Kbit/s}$$

since the portion of traffic due to the BF's false positives is very small compared with true positives.

Nevertheless, as we show in the next section the 1G NetFPGA implementation can handle these values without any problem.

5 Prototype implementation, benchmarking and results

This section introduces the FPGA platform used for the development of our prototype and shows the results of a number of benchmarking tests performed to our monitoring station prototype.

5.1 The NetFPGA platform

Due to its simplicity, versatility, low cost and openness, the NetFPGA [30] framework has been chosen for the implementation of our traffic capture prototype.

The NetFPGA has been developed at Stanford University and provides a basic reference architecture for network hardware implementation, particularly useful in educational and academic environments. There are currently two NetFPGA models operating at different port speeds: 1 Gbit/s and 10 Gbit/s. Both platforms have a stable release. Nevertheless, we have used the 1G NetFPGA for our prototype implementation due to its maturity.

The 1G NetFPGA comes with a Xilinx Virtex II Pro 50 and 4 Gigabit Ethernet copper interfaces. The reference pipeline has a 64-bit word size with a clock speed operating at 125MHz, providing a total raw throughput of 8 Gbit/s. Traffic can be forwarded through the NetFPGA itself and also to/from the server hosting the NetFPGA through a PCI-X bus, allowing the software processing of packets. PCI-X was conceived as an upgrade to prevent certain shortcomings of PCI in servers, as well as improving clock speeds. Nevertheless, in more recent systems, PCI Express (PCIe), with even faster transfer rates, has been the true successor of PCI.

The reference pipeline architecture of the 1G NetFPGA is shown in Fig. 5. This comprises eight reception queues, one Input Arbiter, one Output Port Lookup Module, eight Output Queues and eight Transmission Queues. Concerning the eight transmission and reception queues, four of them belong to the physical ports of the NetFPGA while the other four belong to the virtual ports of the server hosting the FPGA.

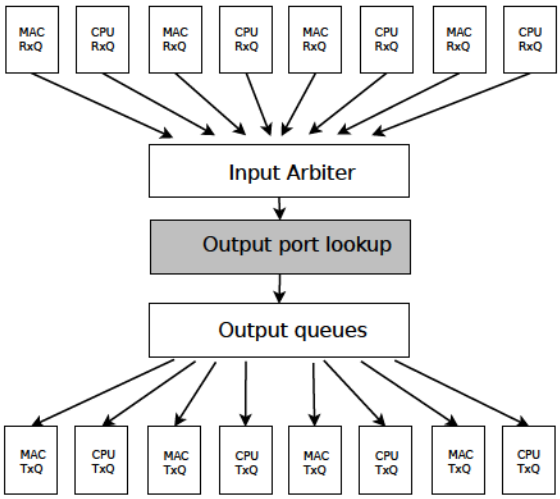


Fig. 5. Reference pipeline architecture of the 1G NetFPGA [30]

Essentially, incoming packets are buffered at the reception queues until the Input Arbiter selects one packet to enter the main pipeline of the NetFPGA. This packet then traverses one or several intermediate modules until it arrives at the Output Queues (see Fig. 5). Finally, this packet can go through zero, one or more Ouput Queues at the same time, depending on certain bits of a special control word used internally by the NetFPGA, thus allowing for traffic replication and multicast.

When developing for the NetFPGA, the usual procedure is to take the reference pipeline as a starting point and either add new modules or substitute old ones on top of it. In this light, our design replaces the Output Port Lookup module of the reference pipeline (shadowed box) by a Bloom-Filter-based packet classification module that selects the suspect's traffic. This is depected in 2.

5.2 Benchmarking scenario

The benchmark scenario can be seen in Fig. 6. The scenario consists of five hosts, four PCs injecting traffic and the server that hosts the NetFPGA. The computer that hosts the FPGA has an Intel Xeon E535 Quad Core CPU running at 2 GHz, with a bus speed of 1333 MHz and 4 GB of DDR2 RAM running at 667 MHz. The other four computers are PCs and servers of heterogeneous features.

To push the monitoring station to its limits, the four PCs connected to our monitoring station inject up to 4 Gbps of traffic. In this configuration, the Tap and Drop mode has been selected with all four ports on the NetFPGA to receive traffic. Four destination IP addresses are trained to a Bloom Filter of size 65536 bits (see section 4.3), one for each connected host. Those packets whose IP address match with any of those trained in the Bloom Filter are copied to the

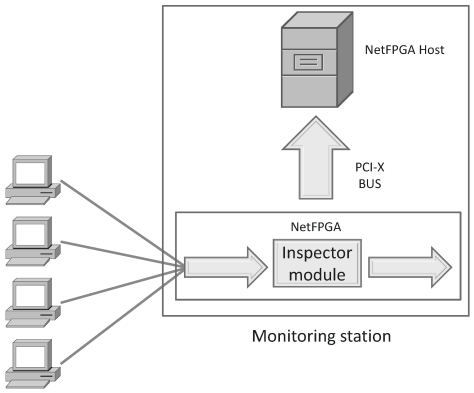


Fig. 6. Benchmarking scenario to evaluate our monitoring station prototype

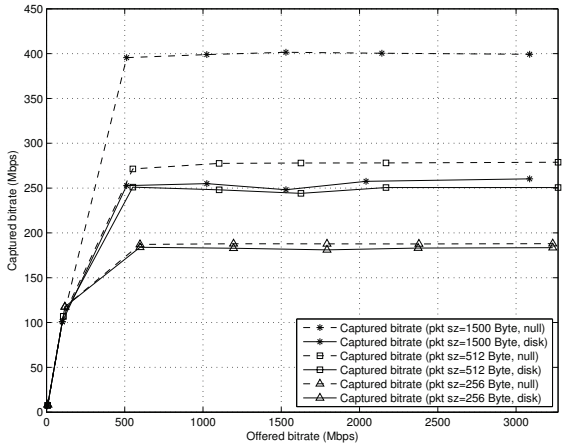
host computer through the PCI-X bus. To run our experiments, several of the connected hosts inject matching traffic at different rates to achieve a total rate inside the NetFPGA pipeline.

Since a certain number of clock cycles is needed to store the packet and check the Bloom Filter, the pipeline efficiency is reduced as the packet size decreases. Consequently, three packet sizes have been used in our tests, namely 256, 512 and 1500 bytes, and two capturing modes to promptly identify the bottlenecks in the PC hosting the NetFPGA. In the first mode, packets are written to a file in the hard disk and in the other the packets are also captured but not written to disk. It is worth noting that, due to limitations in the communication between the host and the FPGA, less traffic than is offered to the device is actually captured by the host. But, according to our measurements, the 4 Gbps injectable to it can be processed by the pipeline, which is plausible, as it is possible to attain a raw 8 Gbps bit rate.

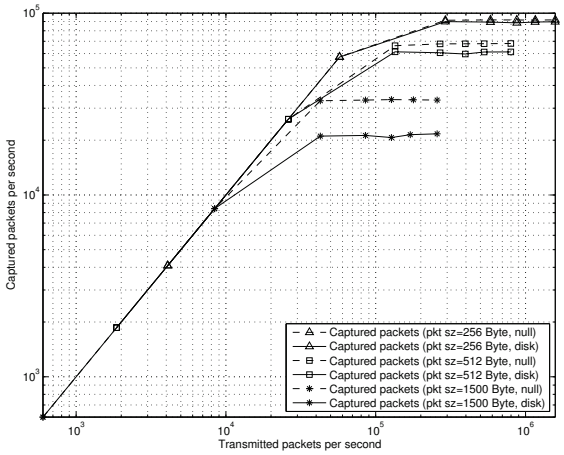
In Fig. 7(a), the captured bit rate vs the offered bitrate for the NetFPGA can be seen. For a packet size of 256 bytes, it can be observed that the main bottleneck is given by the PCI-X bus, conclusion which is reinforced by Fig. 7(b), since the maximum number of packets transferable to the host is achieved either if the packets are sent to disk or not. On the other hand, for a packet size of 1500 bytes, it can be seen that, if packets are not sent to disk, the main bottleneck is given by the PCI-X bus bandwidth (but not the transfer rate, since the number of packets per second is lower than the practical limit achieved as can be observed in Fig. 7(b)). If 1500 bytes packets are written to disk, the bottleneck is clearly given by the hard disk, conclusion which is confirmed by the curve of the bitrate captured to disk for a packet size of 512 bytes, which is quite similar in maximum bandwidth. For a 512 bytes packet size, both curves are not very different, because there are neither rate nor transfer rate limitations in the PCI-X bus. The only limit which makes a significant difference is the hard disk bitrate, hence the distance between both the storing and not storing to disk

curves. From Figs. 7(a) and 7(b) we see that we have a practical transfer rate limit of $9 \cdot 10^4$ captured packets, a limit of 250 Mbps given by the hard disk and a limit of 400 Mbps in the PCI-X bus bit rate.

The CPU is not a bottleneck in any of the cases, since other captures with different Gigabit Ethernet cards have been performed that reached significantly higher rates due to using a different version of the PCI bus.



(a) Captured bit rate vs offered bit rate to the monitoring station



(b) Captured packets per second vs offered packets per second to the monitoring station

Fig. 7. Benchmarking results by bitrate (a) and packets per second (b).

6 Future work and conclusions

In this article we have presented a Bloom Filter-based packet monitoring station for Lawful Interception which we have implemented as a prototype on a 1 Gbps NetFPGA.

It has been shown how Bloom Filters allow high speed filtering with low false positive probability for a reasonable number of users. Furthermore, such claims have been supported with aggregation data from a Spanish ISP to show that our design is scalable to be used in network aggregation points, which would allow Lawful Interception at wirespeed in said aggregation points with reasonable cost per suspect. More importantly, it has been demonstrated that, although traffic capturing in aggregation links could be seen as a tall order, it is not only feasible, but also secure and cheaply realisable if realistic numbers are taken into account to face the problem.

Several traffic capture experiments have been conducted to test the limits of our design. Our design is able to run at wirespeed by injecting traffic at full speed in all ports of the NetFPGA (4x 1Gbps). We have achieved the practical limits of 400 Mbps due to the PCI-X bus of the server, approximately 250 Mbps due to the hard disk, and $9 \cdot 10^4$ captured packets per second due to the transfer limit of the PCI-X bus. Finally, it should be understood that these limitations come from the PC hardware used for the prototype itself (PCI-X bus, Hard disk, etc), but not the NetFPGA itself. It should also be taken into account that our intention, as a future work, is to port this design to a 10G NetFPGA, which might be more suitable for current ISP link capacities.

Acknowledgements

The work presented in this paper has been funded by the INDECT project grant number FP7-ICT-218086, and the Spanish CramNet project (grant no. TEC2012-38362-C03-01).

References

1. A. Broder and M. Mitzenmacher, "Network applications of Bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.
2. Shuai Mu, Xinya Zhang, Nairen Zhang, Jiaxin Lu, Y.S. Deng, and Shu Zhang, "IP routing processing with graphic processors," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, 2010, pp. 93–98.
3. Jin Zhao, Xinya Zhang, Xin Wang, Yangdong Deng, and Xiaoming Fu, "Exploiting graphics processors for high-performance IP lookup in software routers," in *INFOCOM, 2011 Proceedings IEEE*, 2011, pp. 301–305.
4. R. Smith, N. Goyal, J. Ormont, K. Sankaralingam, and C. Estan, "Evaluating GPUs for network packet signature matching," in *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, 2009, pp. 175–184.

5. Lei Wang, Shuhui Chen, Yong Tang, and Jinshu Su, "Gregex: GPU based high speed regular expression matching engine," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, 2011, pp. 366–370.
6. Nen-Fu Huang, Hsien-Wei Hung, Sheng-Hung Lai, Yen-Ming Chu, and Wen-Yen Tsai, "A GPU-based multiple-pattern matching algorithm for network intrusion detection systems," in *Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on*, 2008, pp. 62–67.
7. Cheng-Hung Lin, Chen-Hsiung Liu, and Shih-Chieh Chang, "Accelerating regular expression matching using hierarchical parallel machines on GPU," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, 2011, pp. 1–5.
8. Qiang Wu and T. Wolf, "Runtime task allocation in multicore packet processing systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 10, pp. 1934–1943, 2012.
9. Yunchun Li, Lianqiang Shan, and Xinxin Qiao, "A parallel packet processing runtime system on multi-core network processors," in *Distributed Computing and Applications to Business, Engineering Science (DCABES), 2012 11th International Symposium on*, 2012, pp. 67–71.
10. Y. Yamashita and M. Tsuru, "Rule pattern parallelization of packet filters on muti-core environments," in *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, 2011, pp. 116–125.
11. Danhua Guo, L.N. Bhuyan, and Bin Liu, "An efficient parallelized L7-filter design for multicore servers," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 5, pp. 1426–1439, 2012.
12. "Application Layer Packet Classifier for Linux," 2013.
13. Nen-Fu Huang, Hsien-Wei Hung, and Wen-Yen Tsai, "A unique-pattern based pre-filtering method for rule matching of network security," in *Communications (APCC), 2012 18th Asia-Pacific Conference on*, 2012, pp. 744–748.
14. Haoyu Song, Fang Hao, M. Kodialam, and T. V. Lakshman, "IPv6 lookups using distributed and load balanced bloom filters for 100Gbps core router line cards," in *INFOCOM 2009, IEEE*, 2009, pp. 2518–2526.
15. Sarang Dharmapurikar, Praveen Krishnamurthy, T. Sproull, and J. Lockwood, "Deep packet inspection using parallel Bloom filters," in *High Performance Interconnects, 2003. Proceedings. 11th Symposium on*, 2003, pp. 44–51.
16. Sarang Dharmapurikar, Praveen Krishnamurthy, T.S. Sproull, and J.W. Lockwood, "Deep packet inspection using parallel Bloom filters," *Micro, IEEE*, vol. 24, no. 1, pp. 52–61, 2004.
17. M. Attig, Sarang Dharmapurikar, and J. Lockwood, "Implementation results of Bloom filters for string matching," in *Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on*, 2004, pp. 322–323.
18. M. Attig and J. Lockwood, "SIFT: snort intrusion filter for TCP," in *High Performance Interconnects, 2005. Proceedings. 13th Symposium on*, 2005, pp. 121–127.
19. J. Van Lunteren, "High-performance pattern-matching for intrusion detection," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, 2006, pp. 1–13.
20. N. Tuck, T. Sherwood, B. Calder, and G. Varghese, "Deterministic memory-efficient string matching algorithms for intrusion detection," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 2004, vol. 4, pp. 2628–2639 vol.4.

21. J. Ho and G.G.F. Lemieux, "PERG: A scalable FPGA-based pattern-matching engine with consolidated bloomier filters," in *ICECE Technology, 2008. FPT 2008. International Conference on*, 2008, pp. 73–80.
22. M. Bando, N.S. Artan, Rihua Wei, Xiangyi Guo, and H.J. Chao, "Range hash for regular expression pre-filtering," in *Architectures for Networking and Communications Systems (ANCS), 2010 ACM/IEEE Symposium on*, 2010, pp. 1–12.
23. Y.H. Cho and W.H. Mangione-Smith, "Fast reconfiguring deep packet filter for 1+ gigabit network," in *Field-Programmable Custom Computing Machines, 2005. FCCM 2005. 13th Annual IEEE Symposium on*, 2005, pp. 215–224.
24. R. Ajami and Anh Dinh, "Design a hardware network firewall on FPGA," in *Electrical and Computer Engineering (CCECE), 2011 24th Canadian Conference on*, 2011, pp. 000674–000678.
25. A. Kayssi, L. Harik, R. Ferzli, and M. Fawaz, "FPGA-based internet protocol firewall chip," in *Electronics, Circuits and Systems, 2000. ICECS 2000. The 7th IEEE International Conference on*, 2000, vol. 1, pp. 316–319 vol.1.
26. Sang-Kil Park, Jin-Tae Oh, and Jong-Soo Jang, "High-speed attack mitigation engine by packet filtering and rate-limiting using fpga," in *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, 2006, vol. 1, pp. 6 pp.–685.
27. R. Aparicio, M. Urueña, A. Muñoz, G. Rodríguez, and S. Morcuende, "INDECT Lawful Interception platform: Overview of ILIP decoding and analysis station (accepted for publication)," in *Jornadas de Ingeniera Telemtica (JITEL) 2013*, 2013.
28. M. Urueña, A. Muñoz, R. Aparicio, and G. Rodríguez, "Digital Wiretap Warrant: Protecting civil liberties in ETSI Lawful Interception (review ongoing)," *Computer And Security, Elsevier*.
29. D. Knuth, *The Art of Computer Programming*, vol. 3, Addison-Wesley, 2 edition, 1998.
30. "NetFPGA home page," 2013.