Metroflux: a high performance system for analysing flow at very fine-grain

Patrick Loiseau, Paulo Gonçalves, Romaric Guillier, Matthieu Imbert, Yuetsu Kodama, Pascale Primet Vicat-Blanc

***Abstract*—Researches in network traffic analysis embrace a large diversity of goals and are based on a variety of methodologies and tools. To have a better insight on the real nature and on the evolution of network traffic we argue that fine-grain analysis of real traffic traces have to complement simulations studies as well as coarse grain measurement performed by classical flow measurement systems. In particular, packet level measurements and analysis are needed. However, such methodologies are resource consuming and require very high performance devices to be operational in real high speed networks. In this paper we present the *Metroflux* system which aims at providing researchers and network operators with a very flexible and accurate packet-level traffic analysis toolkit configured for 1 Gbps and 10 Gbps speed links. This system is based on the GtrcNet FPGA-based device technology and on specific statistical analysis tools. We show the potential and the facilities offered by the *Metroflux* system coupled with the *Grid5000* large scale experimental platform and the Network eXperiment Engine (*NXE*) we have developed. We illustrate the application of the *Metroflux* system with the practical validation of the theoretical prediction relating self-similarity and heavy tails given by Taqqu theorem. We also illustrate several usages of this toolset, such as the investigation of conditions under which several traffic theories apply, as well as studies on traffic, protocols and systems interactions.**

I. Motivation

Researches in network traffic analysis aim a large diversity of goals and are based on a variety of methodologies and tools. The experimental environments generally used by researchers are simulators, emulators or production platforms. However, these tools present limitations making some studies difficult and results partial. Simulators focus on a specific behavior or mechanism of the network and abstract the rest of the system. A main restriction of simulators is then the difficulty of their validation and their scale limitation. For example, in an event-driven network simulator, the network is an abstraction and the number of traffic sources or bitrates you can simulate depends on the computing power of the machine executing the simulator. When it becomes difficult to capture and extract the factors influencing the protocols, emulators can help by executing the actual software, in its whole complexity, on a fully controlled platform. As a consequence, there is still a gap between emulators and the reality: they cannot capture all the dynamic, variety and complexity of real life conditions and of real equipments. That is why current Internet links are generally used to observe and analyze the real traffic as it is. The Internet (through projects like PlanetLab, RON...) is also considered as a good candidate for running traffic metrology experiments because it exposes them to realistic conditions. To have a better insight on the real nature and on the evolution of network traffic on these links we claim that fine grain analysis of real traffic have to complement coarse grain measurement performed by classical flow measurement systems. That is notably the case for scaling laws identification in traffic time series. To assess statistical invariance over significantly wide scale ranges, instantaneous throughputs necessarily need to be scrutinized at different aggregation resolutions: from packets inter-arrival time scales up to characteristic coarse scales related to the applications. Furthermore, different scaling properties holding for different observation scales, convey distinct information about the network. Conversely, their influence on the system performances primarily affect mechanisms sensible to the same time scales. For instance, long range dependence (LRD) due to the specific nature of flow size distributions implies time scales that lie beyond the mean flows duration which generally exceeds the buffer's size. As a result, the buffers' load (and the overflows thereof) is more likely to depend on the burstiness of the traffic at fine scales (defined by the RTTs) rather than on the LRD itself. This simple example prompts the need for packet level measurements and analyses. However, such a methodology is resource consuming and requires very high performance devices to be operational in real high speed networks. Indeed emerging experimental research works, exploring for example Future Internet applications based on Data Center and resource sharing are based on very high speed links (10 Gbps). Performing realistic experiments, in order to relevantly characterize the network traffic generated by these applications then requires to solve the challenging issue of monitoring very high speed links at a fine grain (packet level).

On the other hand, while real production link traffic analysis remains a central problem, we also need to perform fully controlled and reproducible experiments which allows the user to vary a lot of parameters (aggregation level, congestion, source flow size distribution, etc.); and then study separately their impact on the traffic. The large scale *Grid5000* instrument

Patrick Loiseau is with Université de Lyon, École Normale Supérieure de Lyon (LIP), France. (e-mail: Patrick.Loiseau@ens-lyon.fr)

Paulo Gonçalves, Romaric Guillier, Matthieu Imbert and Pascale Vicat-Blanc Primet are with INRIA, Université de Lyon, École Normale Supérieure de Lyon (LIP), France. (e-mails: Paulo.Goncalves@ens-lyon.fr, Romaric.Guillier@ens-lyon.fr, Matthieu.Imbert@ens-lyon.fr, Pascale.Primet@ens-lyon.fr)

Yuetsu Kodama is with National Institute of Advanced Industrial Science and Technology (AIST), Japan. (e-mail: y-kodama@aist.go.jp)

providing up to 5000 fully reservable and reconfigurable cores combined with the *Metroflux* system offers such reproducible environment.

In this paper, we present *Metroflux*, a fully operational metrology platform, based on the GtrcNet-1 and GtrcNET-10 hardware, which enables to capture the traffic at packet level on a very high speed link and to analyze it at packet and flow level. The *Metroflux* system aims at providing researchers and network operators with a very flexible and accurate packet-level traffic analysis toolkit configured for 1 Gbps and 10 Gbps links. This system is based on the GtrcNet FPGA-based device technology and original statistical analysis tools. We then present its utilization in two different situations: a controlled experiment on *Grid5000* and the monitoring of a production link. Firstly, we illustrate the application of the *Metroflux* system with the experimental validation of the theoretical prediction relating self-similarity and heavy tails given by Taqqu theorem. We also show how the potential and facilities offered by the *Grid5000* instrument and *Metroflux* system enable to investigate the conditions under which several traffic theories apply as well as to better understand how traffic, protocols and systems interplay. The paper is organised as follows. Section II presents the general design of our plateform. In Section III two use cases are detailed to illustrate the potential of *Metroflux* system. Related works are reviewed in Section IV. Finally, we conclude in Section V.

II. General design of the *Metroflux* system

A. Global architecture of the system

Metroflux is a programmable system for flow analysis which currently operates on a 1 Gbps link without loss, and which is in a validation phase for 10 Gbps links. As described in figure 1, this system is composed of hardware elements and software components. The *Metroflux* system integrates the GtrcNet-1 box [1] or the GtrcNet-10 box [2] (for 10 Gbps links) and a storage server with large amount of disk space. The system is able to capture the first 52 bytes (for GtrcNET-1) or 56 bytes (for GtrcNET-10) of every packets (*i.e.* the header and few payload bytes), group them and send them to a capture server. The capture server runs a MAPI (Monitoring API [3], developed by the LOBSTER project [4]) daemon with a special GtrcNET driver, and is able to save the packet headers' stream into a pcap file for offline analysis. The pcap format has been choosen because it is also used by tcpdump and it saves packets with their capturing timestamp. The data analysis server runs a packet header extracting program, which is based on the MAPI library which provides advanced functions to manipulate packet headers in pcap format. The duration of a capture session depends on the packet size and the throughput of the link. It can encompass several hours in low load conditions.

The *Metroflux* system can be installed transparently within an experimental or a production network in two different ways (see Figure 2):

- incorporated in a 1 Gbps links (between source(s) and destination(s)),

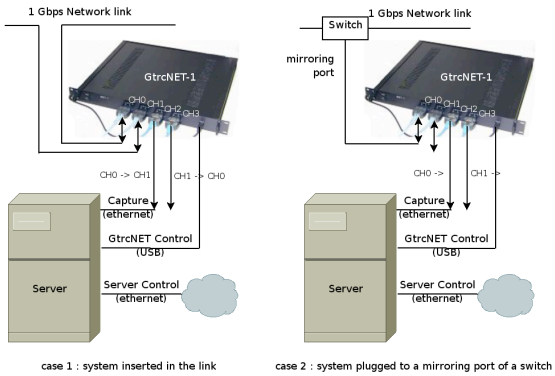

Fig. 2. Two different ways of installing *Metroflux* on a link

- plugged into a mirrored port of a switch.

If the *Metroflux* system is inserted in the data path, other GtrcNET functions such as latency emulation or aggregated throughput on-line measurement can be activated.

In next Section, we develop the specificities of the GtrcNET-10.

B. GtrcNET-10 characteristics for 10 Gbps links measurement

GtrcNET-10 [2] is a hardware layer 2 network equipment with 10 GbE ports, and it is a successor of GtrcNET-1 [1] with GbE ports. GtrcNET-10 provides many parametrable functions, such as traffic monitoring in microsecond resolution, traffic shaping, and WAN emulation at 10 Gbps wire speed. A remote computer can set several parameters, such as interval of traffic bandwidth monitoring, target bandwidth of traffic shaping, and delay of network emulation. It gets results of bandwidth monitoring. In the *Metroflux* system, GtrcNET-10 is used and configured to capture headers of packets. GtrcNET-10 consists of a large-scale Field Programmable Gate Array (FPGA), three 10 Gbps Ethernet XENPAK ports, and three blocks of 1 GB DDR-SDRAM. Figure 3 shows the architecture of GtrcNET-10. The FPGA is a Xilinx XC2VP100, which includes three 10 Gbps Ethernet MAC and XAUI interfaces. The FPGA can directly receive the 3.125 GHz signals from XENPAK module by Rocket I/O hardware macro. The main circuit in FPGA runs on 156.25 MHz clock with 64 bit data width, and the memory is accessed by 162 MHz to support read and write with wire rate speed of 10 GbE. GtrcNET-10 is connected to a control PC via USB. It also has a serial port to connect GPS module, and it can synchronize the time with other GtrcNET by receiving accurate time from GPS. The MICTOR connector is used for debugging in the circuit of FPGA. System ACE initializes the FPGA by the configuration data on Compact Flush memory.

By programming the FPGA, one can add new functions and improve existing functions according to the requirements of the users.

Let us now detail packet capture functions of GtrcNET-10 with two examples.

Figure 4 (a) shows an example of usage for packet capture on GtrcNET-10. The traffic to be captured is transferred

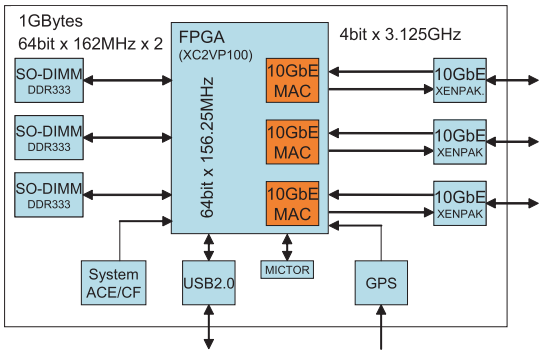

Fig. 3. The architecture of GtrcNET-10

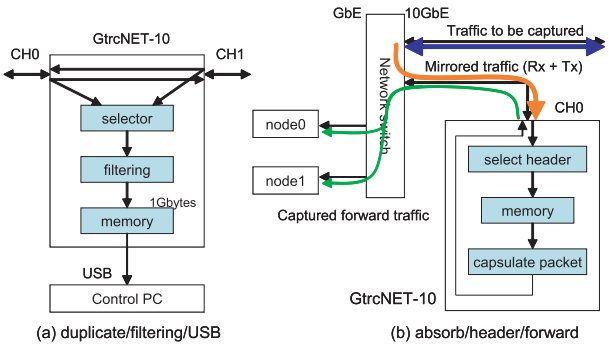

Fig. 4. Example of packet capture on GtrcNET-10

between ports CH0 and CH1. GtrcNET-10 selects an input port, duplicates packets from the port, and captures them into a memory. This function cannot capture bi-directional traffic. The traffic transfered between ports CH0 and CH1 is not affected by the capture. A filtering condition is defined so that packets are captured only if they satisfy the filtering conditions. A condition indicates a 16 bit field of the header, selects any bits by 16 bit mask, and compares it with the specified 16 bit value. One or two conditions can be specified, and the logical combination of two conditions (logical-and or logical-or) can also be specified. For example, one can capture only packets whose source port or destination port of TCP/IP header coincides with a specified value. One can also capture only packets that have VLAN tag, the VLAN ID being a specified value. The captured data are sent to a control PC via USB. Since the USB access speed of current implementation is only 100 Kbytes/sec, the capture size is limited to the memory size, which is 1 Gbytes. If all the packets of a wire rate traffic at a 10 GbE link are captured, only 800 ms of traffic can be captured.

Figure 4 (b) shows another example of usage for packet capture on GtrcNET-10. The traffic to be captured is mirrored by a network switch, and the mirrored traffic is transferred to GtrcNET-10. GtrcNET-10 can capture only selected header fields of the packets into a memory. The header fields to be captured are in 16 bytes unit long, and any field can be selected independently up to 128 bytes. The received timestamp is a

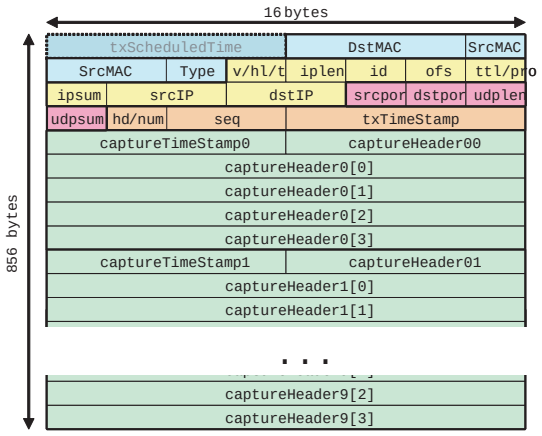

Fig. 5. Packet format of capture forward

64 bit field whose format is based on RFC-1305 [5] and it constitutes the first 8 bytes of the captured header fields for each packet. Multiple captured headers are encapsulated into a UDP packet, and transferred from an output port. Figure 5 shows the format of the UDP packet, which includes ten 80 bytes of captured data. The first block is a scheduled time for transmission. This field is only used inside of GtrcNET-10, and is not transmitted. The next three blocks are Ethernet, IP, and UDP headers respectively. The fifth block is a header of the forward packet that includes captured header size (hd), number of captured headers (num), sequence number of packet (seq), and its transmit timestamp (txTimestamp). Captured data follow the headers. If the one-way wire rate traffic of a 10 GbE link with 1500 bytes IP packet is captured using the format in Figure 5, the capture forward traffic rate is about 556 Mbps. The forward traffic should be read by a receiving node. If the traffic is too large to be read by single node, the destination of the forward traffic can be distributed to multiple nodes in round-robin, up to 16 nodes.

By mixing received and transmitted traffic in the mirrored traffic, bi-directional traffic can be captured in a port, but the mirrored traffic is limited to 10 Gbps. Since a GtrcNET-10 can capture mirrored packets from three ports simultaneously, bi-directional traffic of wire rate can be captured by mirroring received and transmitted traffic independently. Notice that some switches cannot receive packets from the mirroring port. Since GtrcNET-10 can transmit capture forward packet from any port, it solves the problem, but the solution requires two ports of GtrcNET-10 for packet capturing of one port.

GtrcNET-10 can capture packets in any combination, such as duplicating packets or not, filtering packets or not, selecting header fields or not, accessing by USB or forwarding as packets, but capture forwarding is only for selected headers.

In the *Metroflux* system, GtrcNET-10 is used with a server which stores the packet headers' stream and perform off-line fine grain traffic analysis.

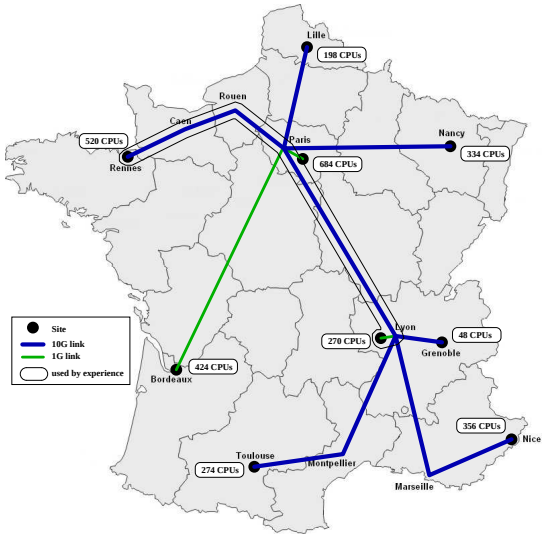

Fig. 6. *Grid5000*

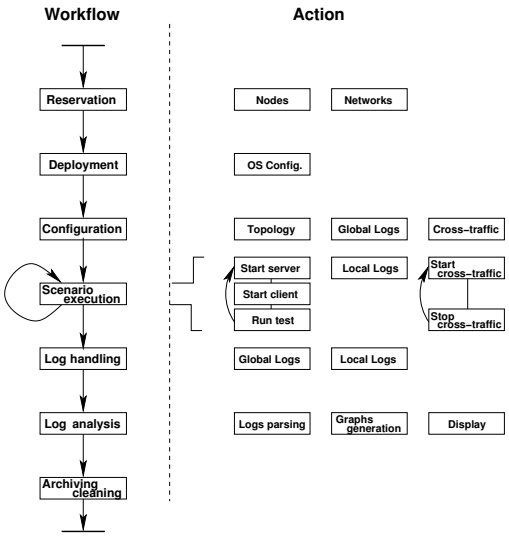

Fig. 7. Workflow used by the NXE tool

C. *Integration of* Metroflux *in a controlled environment*

The *Metroflux* system has been adapted to instrument the fully controlled environment *Grid5000*. In addition, an original and extensible tool to design a traffic analysis experiment and to automatically deploy it has been written [6].

Traditional production networks hardly support traffic analysis experimentation for studying the relationship between traffic patterns, aggregate characteristics and protocols. Indeed, reproducing the experimental conditions several times is almost impossible in shared and uncontrolled networks. Moreover, such experiments require a large scale deployment of end-host protocols and mechanisms. These software (or kernel modules) like rate limitation, QoS mechanisms, congestion control variants should be implemented within the operating system of communicating nodes. The experiments require the installation of experimental hardware and traffic capture elements in a representative traffic aggregation point. This type of experiments is difficult to run in production environments. Other experiment classes are concerned by such limitations, like for example experiments which exploit high capacities and reconfigurable networks, which need programmable network equipments, which need information services not exposed within the current Internet, which need to be deployed on fully reservable and configurable computing or switching fabrics, which may congest links and present security risks. The national *Grid5000* plate-forme has been designed and built to offer researchers such a large-scale realistic and reconfigurable instrument.

Grid5000 is a research tool, featured with deep control, reconfiguration and monitoring capabilities to complement network simulators and emulators. It allows the users to reserve the same set of dedicated nodes across successive experiments, and to have full controll of these nodes to run their own experimental condition injector and measurement software. This can be achieve thanks to two tools: OAR and Kadeploy. OAR is a reservation tool which offers advanced features (CPU/Core/Switch reservation). Kadeploy is an environment deployment system which allows the users to have their own customized environment automatically deployed on a large number of nodes. For example, kernel modules for rate limitation, congestion control variants or QoS measurement can be added to the native operative system.

Grid5000 is a 5000 CPUs nation wide grid infrastructure dedicated to network and grid computing research [7]. Up to 17 French laboratories are involved and 9 sites geographically distributed are hosting one or more cluster of about 500 cores each (Figure 6). The sites are interconnected by a dedicated optical network provided by RENATER, the French National Research and Education Network. It is composed of private 10 Gbps Ethernet links connected to a DWDM core with dedicated 10 Gbps lambdas, with a bottleneck at 1 Gbps in Bordeaux. Two international interconnections are also available: one at 10 Gbps with DAS3 (the Netherlands) and one at 1 Gbps with Nareji (Japan). As a private testbed dedicated to research, *Grid5000* made it easy to install the experimental hardware of *Metroflux* at representative traffic aggregation points. In the scenario detailled below, the *Metroflux* system is measuring the access link of the *Grid5000* Lyon site.

D. *Network eXperiment Engine*

The Network eXperiment Engine (*NXE*) [6] is a tool to automate networking experiments in real testbeds environments. It allows to simply script experiments involving hundreds of nodes.

A networking experiment is described with a scenario skeleton defined as a succession of dates at which events occur. An event corresponds to the starting point of an action (*e.g.* the start of a new bulk data transfer, of a new web session) combined with the parameters relevant to this action (*e.g.* distribution law of file sizes, inter waits). An action takes place between a set of end-hosts, whose size depends on the

kind of application we are trying to model (*e.g.* 2 for data transfers, many for parallel applications).

The end-hosts are organised in a networking abstract topology, that roughly defines sites (*e.g.* aggregation of end-hosts, as in a cluster), aggregation points between them (*e.g.* switches or routers) and networking links. The "abstract" term refers to the fact that an instantiation of the nodes over this topology is needed at run-time as in real testbeds, and resources allocation mechanisms might be used to accommodate multiple users.

Figure 7 shows the experiment workflow, and the various operations that are done at each stage of the execution of a protocol evaluation scenario. This workflow is a description of an evaluation process. It is composed of a number of tasks which are connected in the form of a directed graph. These tasks have been broken up into elementary operations to explicit what is done precisely at each stage. The tasks were designed so that there is as little interaction as possible between successive tasks. The description of each stage of the workflow is as follows:

Reservation: at this stage, the available resource allocator services are contacted to get the resources needed by the experiment, *e.g.* computing nodes or network links.

Deployment: this configuration phase can be either a reboot of the nodes using an adequate kernel image, or just the setting of the OS internal variables (*e.g.* TCP buffer size) to the appropriate value.

Configuration: at this stage, the available hardware (*e.g.* hardware latency emulator, routers) are contacted to alter the topology, or to activate the gathering of statistical information (*e.g.* aggregate throughput) according to the needs of the experiment.

Scenario execution: here the actual execution of the scenario is started. The scenario can be run multiple times in a row to ensure that the results are consistent.

Log handling: the logs generated by the nodes and the global logging facility are gathered at a single point for analysis.

Log analysis: the logs are parsed, and metrics are computed from them to generate graphs that can be easily interpreted by the user.

Archiving and cleaning: resources are reset and released.

The scenarios are described through XML files that provide a simple and hierarchical description of the topology, the general configuration and the interactions between the end-hosts.

NXE is an application that scripts the execution of a schedule based on a centralised relative timer, that is generated from the input scenario description. It assumes that the granularity of the scenario execution steps is coarse and in the same order of magnitude as a second. For scalability purposes, it launches a separate thread for every node involved in the scenario, and issues the commands at the appropriate time via an SSH remote command execution. Only one SSH connection is opened per node

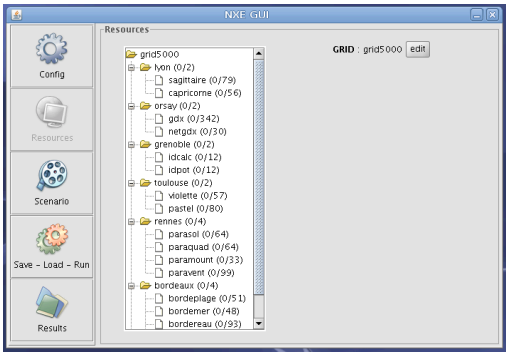

Fig. 8. Screenshot of the NXE GUI, resource selection part

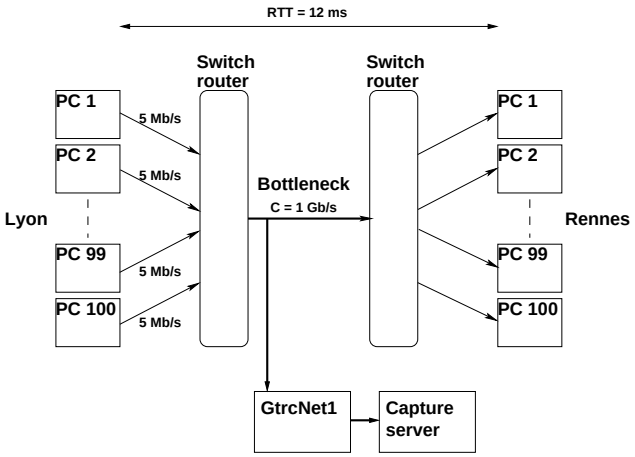

Fig. 9. Experimental topology

Figure 8 presents the graphical interface that was developped over the NXE software to automatically generate the input XML files. Here the resource selection part over *Grid5000* is featured. It also allows the simple definition of the timeline of a scenario.

More information can be found at the following website address: http://www.ens-lyon.fr/LIP/RESO/Software/NXE/index.html

III. Use cases

We now present two typical examples that clearly illustrate the assets of our experimental facility.

A. Measurement and traffic analysis of a controlled experiment

The aim of this experiment is to validate on a large scale experimental network the relationship between the tail index of the (heavy-tailed) flow size distribution and the long-range dependence (LRD) parameter of the aggregated traffic [8], [9]. This experiment was initially performed with a 1 Gbps bottleneck with the GtrcNET-1. The method remains the same when using a GtrcNET-10 on a 10 Gbps link.

1) Scenario description: The topology used for this experiment is described on Figure 9: 100 clients nodes in Lyon (sources) are emitting to 100 server nodes in Rennes (destinations). The average RTT is 12 ms. Each client behaves like a ON/OFF source: ON periods correspond to a flow emission and OFF periods to an idle time separating two consecutive ON periods. The source rate during an ON period is limited to 5 Mbps to avoid congestion at the 1 Gbps bottleneck. The rate limitation mechanism can be chosen: pspacer, token bucket or TCP window limitation.

For the purpose of our experiment, the ON periods are independent and identically distributed random variables, following a heavy-tailed distribution of tail index α. The OFF periods are exponentially distributed with the same mean as the ON periods.

During the 8-hours experiment, the out-going traffic from Lyon is mirrored and sent to the capture device (see Figure 9). It results in a trace stored on the server, containing all the packet headers with their associated timestamp packets, that can be used for off-line statistical analysis.

2) Analysis and results from the captured trace: All the informations needed to compute the tail index and LRD index can now be retrieved from the trace.

Firstly, grouping and counting the packets in each contiguous time interval yields the aggregate traffic time series, from which the LRD index can be measured using appropriate wavelet-based tools [10].

Secondly, flows can be reconstructed from the trace. To do so, the packets sharing the same IP source and destination addresses, the same source and destination ports and the same protocol are grouped into a flow, as long as two such consecutive packets are not separated by more than a threshold named `timeout`, whose value has to be carefully chosen. From this flow reconstruction, we then can easily extract the flow sizes, and then estimate the tail index to verify that it coincides with that imposed to the ON duration distribution. The tail index estimator used is a recent wavelet-based estimator whose good performances are shown in [11].

Thus, from a trace capture at an aggregated link, we are able to measure the tail index and the LRD index, and then to check the link between these two indices. The use of the fully controllable tool *Grid5000* allows us to perform the same experiment many times, while varying a lot of parameters: the protocol, the tail index, the rate limitation, etc. The impact of all these parameters can then be thoroughly studied. More details about this kind of experiments, and the results can be found in [12].

3) Sampling: Even if GtrcNET-10 enables full packet capture at 10 Gbps, the capture of all the timestamped packet headers at this speed can prove very demanding in terms of storage resources. To limit this resource consumption, a sampling function has been implemented in GtrcNET-10: when activated, it captures only one packet every N packets, for an arbitrary value of N.

Yet, because of its aggregated characteristic, the LRD parameter can still easily be estimated from a sub-sampled trace, whereas the estimation of the tail index becomes much more complicated in such situation. To solve this problem, we have derived in [13] the exact maximum likelihood solution for the estimation of the tail index from sub-sampled data.

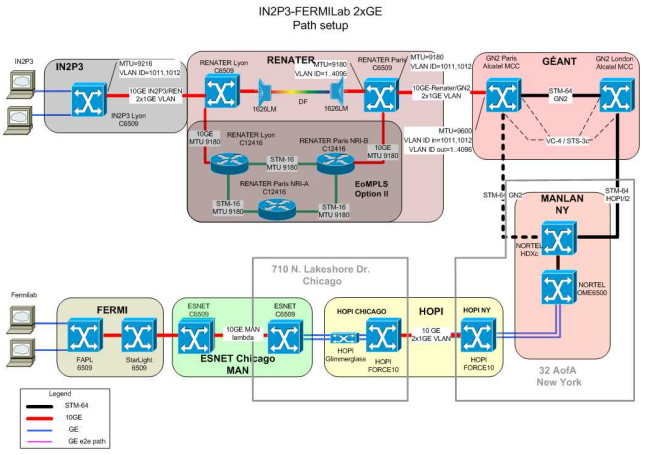

Fig. 10. in2p3-fermilab link

B. Measurement and analysis of real production grid traffic

1) Description of the analyzed link: Figure 10 shows the paths of the production traffic between in2p3 (Lyon, France) and fermilab (Chicago, IL, USA). We "plugged" *Metroflux* at the output of the in2p3 to capture the traffic of the 1 Gbps Ethernet VLAN encapsulated in the 10 Gbps link, using a GtrcNET-1.

2) Results: We now present the results obtained from a 50 days continuous capture on the link described in the previous subsection. Although we were able to monitor outgoing traffic as well as incoming traffic, we only present the results associated with incoming traffic (both have the same characteristics).

As explained in Example I, we are able to reconstruct from the capture the list of all the flows with their characteristics (length (in packets), volume (in Bytes), duration, etc.). Figure 11 shows the distribution of the flow volume and duration, where we separated three different types of flows with respect to the mean size μ of the flow's packets: $\mu = 64$ (small packets, mainly ACKs' flows), $\mu = 1448$ (large packets, typical of data's flows), $64 < \mu < 1448$ (intermediate). Interestingly, we can see that the tail of the flow volume distribution is constituted by flows of large packets (blue curve), but the tail of the flow duration distribution is constituted by flows of small and intermediate packets (red and green curves). It means that the largest flows in term of volume are not the longest flows in term of duration.

To complement this study with a joint analysis of the duration and the volume, Figure 12 represents the flow duration against the flow volume for the three flow types, and for all the flows. We see again in Figure 12 that the longest (in term of duration) flows are not the largest ones (in term of volume).

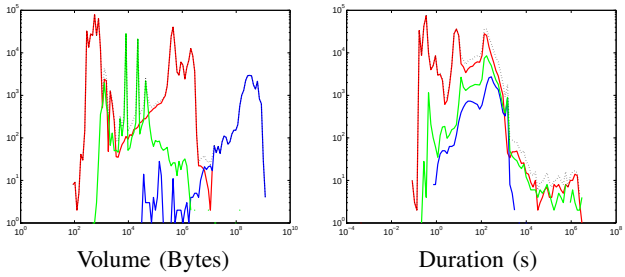

Fig. 11. Distributions (loglog plot) of the flow volume and duration for different mean packet size. (μ, in Bytes): (red) $\mu = 64$ – (blue) $\mu = 1448$ – (green) $64 < \mu < 1448$

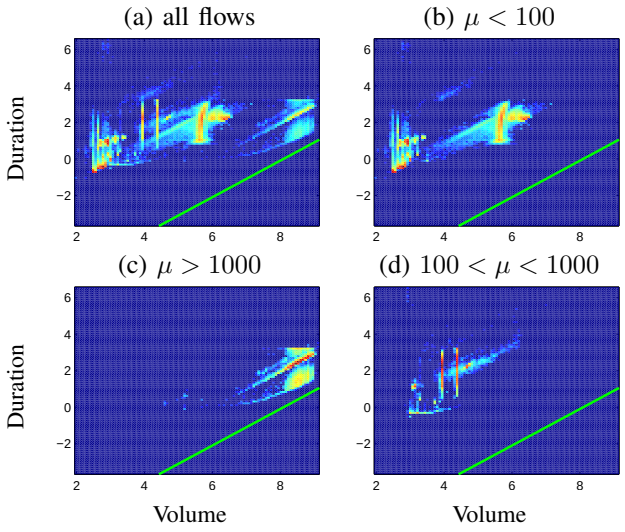

Fig. 12. Flow duration against flow volume for different mean packet sizes. The line is the minimal flow duration for a given volume (at 1 Gbps).

IV. Related work

A few studies have already focused on the feasibilty of network traffic capture at speeds as high as 10 Gbps, using commodity hardware ([14], [15]). Most of these studies and of the few solutions available on the market to perform traffic capture (for example, products from Endace, Solera Networks, or GigaStor) focus on full traffic capture, whereas our requirements only include the capture of packet headers, which are sufficient for our needs. Besides, a lot of the solutions available today are designed and marketed as intrustion detection or prevention systems, and focus on real-time deep-packet inspection rather than traffic capture.

Moreover, our requirements also include the capability of capturing headers with no loss, or very few packet loss even in the worst case of very small packets arriving at the maximal load. In this worst case, the capture system has to cope with a data bandwith which is almost as high as if it was storing the full packets instead of the headers, so that the technical problems are quite similar:

1) The capture server has to handle a high number of packets per second, which will in turn trigger a high number of interrupts per second, if using the legacy network stack.
2) The data path in the capture system has to be as short and direct as possible, and keep data copy to a minimum.
3) The disk storage has to be both very large and very fast.

In [14], the authors come to the conclusion that commodity hardware is unable to fulfill all these requirements and they propose to distribute to capture among several servers. DAG cards, from Endace, which are quite popular in the research community, address the first two problems by optimizing data path in the host system and providing an appropriate API. The authors in [15] solve the two first problems by using a state of the art (at the time of the writing) high performance server, carrefully tweaked for the task, and custom software optimizing the capture. They solve the storage problem by using a very powerfull raid array. They also use a special dataseries file format for the storage.

In our solution, the first problem is solved by the fact that the GtrcNET groups a lot of headers in a UDP frame (currently: 62 headers per frame with GtrcNET-10). The number of interrupts is therefore divided by 62. The second problem is addressed by using the MAPI library with a custom GtrcNET driver (using libpcap internally) and some optimizations. This is perhaps not the optimal solution, but it has proved to be effective. Finally, thanks to the continuous increase in hardware performances, we were able to prepare the capture server and disk storage enclosure using very common hardware, at a very competitive cost. The storage system is made of 15 disks organized in a RAID-0 array. These disks are 300 GB 15K RPM SAS disks, for a total of 4.5 TB. At some point during the project we also evaluated the possibility to use Solid State Storages instead of hard disk storage, but this technology is still very new.

Finally, being based on a programmable FPGA network appliance separated from the capture server, *Metroflux* allows a lot of processing to be performed in hardware, including filtering and/or sampling of traffic to capture. A lot of other functions are also available (traffic shaping, latency emulation, custom functions).

V. Conclusion

In this paper we have presented *Metroflux*, a fully operational metrology platform based on the GtrcNET hardware. It enables a full packet header capture at very high speed links (10 Gbps) and a fine grain flow analysis. In two examples, we have shown how *Metroflux* can be used to monitor high speed links at packet level and get usefull insights on the characteristics of the traffic going through these links. It makes from *Metroflux* an original monitoring tool of primary importance for very high speed network research. Then we plan to deploy several Metroflux system in each *Grid5000* site. Our future work is to develop an interface and a software library for easily accessing Metroflux to provide researchers with a customizable tool for debugging their application and better understand how the traffic behaves and interacts with their own flows during an experiment on the *Grid5000* testbed or other research facility.

VI. Acknowledgement

This work has been funded by the French ministry of Education and Research, INRIA, and CNRS, via ACI GRID's Grid'5000 project , the ANR IGTMD grant, IST EC-GIN project, INRIA GridNet-FJ grant, NEGST CNRS-JSP project.

References

[1] Y. Kodama, T. Kudoh, R. Takano, H. Sato, O. Tatebe, and S. Sekiguchi, "Gnet-1: Gigabit ethernet network testbed," in *Proc. of 2004 IEEE International Conference on Cluster Computing (Cluster2004)*, 2004, pp. 185 – 192.

[2] http://www.gtrc.aist.go.jp/gnet.

[3] M. Polychronakis, K. G. Anagnostakis, E. P. Markatos, and A. ysleb, "Design of an application programming interface for ip network monitoring," in *Proc. of the 9th IFIP/IEEE Network Operations and Management Symposium (NOMS04)*, April 2004, pp. 483–496. [Online]. Available: http://mapi.uninett.no

[4] http://www.ist-lobster.org/.

[5] D. L. Mills, "Network time protocol (version 3) specification, implementation and analysis," RFC 1035, March 1992.

[6] R. Guillier and P. Vicat-Blanc Primet, "Methodologies and tools for exploring transport protocols in the context of high-speed networks," in *IEEE TCSC Doctoral Symposium*, May 2008.

[7] R. Bolze, F. Cappello, E. Caron, M. Daydé, F. Desprez, E. Jeannot, Y. Jégou, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, B. Quetier, O. Richard, E.-G. Talbi, and I. Touche, "Grid'5000: a large scale and highly reconfigurable experimental grid testbed." *Int. J. of High Performance Computing Applications*, vol. 20, no. 4, pp. 481–494, nov 2006.

[8] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic (extended version)," *ACM/IEEE Trans. on Net.*, vol. 2, no. 1, pp. 1–15, Feb. 1994.

[9] M. S. Taqqu, W. Willinger, and R. Sherman, "Proof of a fundamental result in self-similar traffic modeling," *SIGCOMM CCR*, vol. 27, no. 2, pp. 5–23, 1997.

[10] P. Abry, P. Flandrin, M. Taqqu, and D. Veitch, "Wavelets for the analysis, estimation and synthesis of scaling data," in *Self-Similar Network Traffic and Performance Evaluation*, K. Park and W. Willinger, Eds. John Wiley & Sons, Inc., 2000.

[11] P. Gonçalves and R. Riedi, "Diverging moments and parameter estimation," *J. of American Stat. Asso.*, vol. 100, no. 472, pp. 1382–1393, December 2005.

[12] P. Loiseau, P. Gonçalves, G. Dewaele, P. Borgnat, P. Abry, , and P. V.-B. Primet, "Investigating self-similarity and heavy-tailed distributions on a large scale experimental facility," INRIA, Tech. Rep. 6472, March 2008.

[13] P. Loiseau, P. Gonçalves, S. Girard, F. Forbes, and P. Primet Vicat-Blanc, "Maximum likelihood estimation of the flow size distribution tail index from sampled data," September 2008, preprint.

[14] F. Schneider, J. Wallerich, and A. Feldmann, "Packet Capture in 10-Gigabit Ethernet Environments Using Contemporary Commodity Hardware," in *Passive and Active Network Measurement: 8th International Conference, Pam 2007, Louvain-la-neuve, Belgium, April 5-6, 2007, Proceedings*. Springer, 2007.

[15] E. Anderson and M. Arlitt, "Full packet capture and offline analysis on 1 and 10 gb networks," HP, Tech. Rep. HPL-2006-156, 2006.

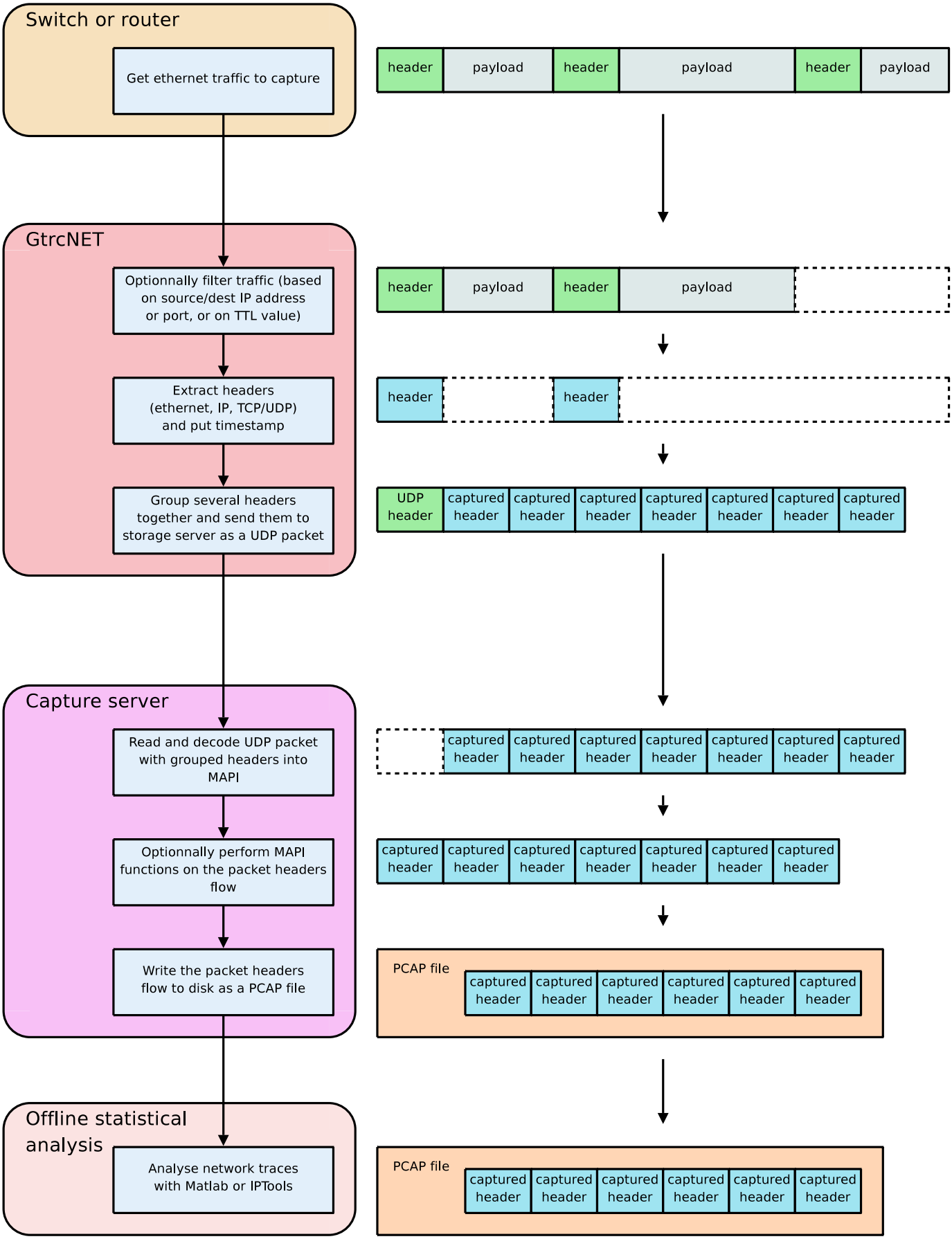

Fig. 1. Capture and analysis of the traffic with *Metroflux*