Increasing Data Center Network Visibility with Cisco NetFlow-Lite

Luca Deri
ntop, IIT-CNR
Pisa, Italy
deri@ntop.org

Ellie Chou, Zach Cherian, Kedar Karmarkar
Cisco Systems
San Jose, CA, USA
{wjchou, zcherian, kedark}@cisco.com

Mike Patterson
Plixer Inc
Sanford, ME, USA
mike@plixer.com

Abstract— NetFlow is the de-facto protocol used to collect IP traffic information by categorizing packets in flows and obtain important flow information, such as IP address, TCP/UDP ports, byte counts. With information obtained from NetFlow, IT managers can gain insights into the activities in the network. NetFlow has become a key tool for network troubleshooting, capacity planning, and anomaly detection. Due to its nature to examine every packet, NetFlow is often implemented on expensive custom ASIC or else suffer major performance hit for packet forwarding, thus limit the adoption. NetFlow-Lite bridges the gap as a lower-cost solution, providing the network visibility similar to those delivered by NetFlow.

This paper describes the architecture and implementation of NetFlow-Lite, and how it integrates with nProbe to provide a scalable and easy-to-adopt solution. The validation phase carried on Catalyst 4948E switches has demonstrated that NetFlow-Lite can efficiently monitor high-speed networks and deliver results similar to those provided by NetFlow with satisfactory accuracy.

Keywords-component; Passive traffic monitoring, NetFlow-Lite.

I. INTRODUCTION AND MOTIVATION

NetFlow [1] and IPFIX are two popular traffic monitoring protocols that allow to classify traffic in flows. Within this context, a flow is defined [2] as a set of IP packets passing through an observation point during a certain time interval. Packets belonging to a flow have a set of common header properties including IP/port source/destination, VLAN, application protocol and TOS (Type of Service). In both NetFlow and IPFIX the flow probe, responsible for aggregating packets into flows, is usually embedded into the networks device where flows the traffic to be analyzed. When traffic analysis capabilities are missing from the network devices, it is also possible to export packets (e.g. using a span port or a network tap) from the network device to a PC and run let them be analyzed by a software probe running on PCs [4] [5].

When flows are expired, either due to timeout or maximum duration, they are exported out of the device to a flow collector via UDP/SCTP formatted in NetFlow/IPFIX format. The flow collector usually runs on a PC, and it often dumps flows on a database after flow filtering and aggregation. Unlike SNMP [3], NetFlow/IPFIX are based on the push paradigm where the probe sends flows to the collector, without allowing the collector to periodically read flows from the probe.

As flows are computed on IP packets, thus limiting NetFlow/IPFIX visibility to the IP protocol. Although flow-based analysis is quite accurate, it is relatively heavy for the probe as every packet need to be decoded and also because the number of active flows increases with the traffic rate. In order to cope with high-speed traffic analysis while preventing NetFlow/IPFIX to take over all the available resources on the monitoring device, often sampling techniques are used [10]. Sampling can both happen at packet [6] and flow [7] level. In the former case reducing the amount of traffic to be analyzed also reduces the load on the probe, but often not the number of flows being computed; in the latter case, reducing the number of exported flows decreases the load on the collector with little relief on the probe side. Unfortunately the use of sampling leads to inaccuracy [8] [9], and thus network operators prefer to avoid it if possible.

Although on layer-three routers the use of sampling is not desirable, monitoring high-speed switches without sampling is not really feasible. This is because the total aggregate port traffic can very well exceed 100 Gbit (if not 1 Tbit), thus either monitoring is restricted to a limited set of ports or some packet sampling techniques have to be used. Furthermore it is a common misconception that sampling reduces accuracy of measurements [11].

Motivation

In today's complex network environment, applications with diverse purposes converge on common network infrastructure, users from different geographic locations connect to the same physical network through different methods. As a result of that, having the visibility into the network activities and application traffic is critical to many IT managers.

For years people have been using NetFlow to gain insight into the network traffic. However, NetFlow is not always an available option. In some places in network, the networking gear is often not equipped with such capability due to the architecture design and cost structure to fit into that specific market, for example data center ToR switches.

Flexible NetFlow is an evolution of NetFlow. It utilizes the extensible format of NetFlow version 9 or IPFIX and has the

ability to export not only the key fields seen in traditional NetFlow, but also the new fields such as packet section. Flexible NetFlow also introduces the concept of immediate cache which allows immediate export of flow information without hosting a local cache. NetFlow-lite [13] is built upon the flexibility of Flexible NetFlow, with the combination of packet sampling, to offer the visibility similar to those delivered by NetFlow at a lower price point, without the use of expensive customer ASIC while maintaining the packet forwarding performance.

Due to the pervasiveness of NetFlow in many parts of the network, the solution also needs to be designed to integrate easily with existing infrastructure that is already monitoring through NetFlow. In addition, the solution needs to be scalable in order to accommodate the rapid growth of today's network, especially in mega-scale data centers (MSDCs), where thousands of servers are connected to provide the application services to scale to the business needs. One challenge that arises when monitoring networking devices with a centralized collector/analyzer is the extra amount of traffic it generates and traverses through the network. Not only does valuable bandwidth being taken up, but also the centralized collector might not be able to scale up to meet the demands.

This is where the NetFlow-lite converter, such as nProbe, fits in. It bridges the world between NetFlow-lite and NetFlow. It parses the packet section exported through NetFlow version 9 or IPFIX format, extracts key information such as src/dst IP address, TCP/UDP port, packet length, etc., it constructs temporary flow cache, extrapolate flow statistics by correlating sampling rate w/ sampled packets, exports aggregated and extrapolated data to NetFlow collectors in standard IPFIX or NetFlow v5/v9 format. With this solution, the valuable forwarding bandwidth is conserved by aggregating NetFlow-lite data to more bandwidth efficient NetFlow export

In a nutshell, NetFlow-Lite is a technology that provides visibility in the data center as it enables network administrators to:

- Know what applications are consuming bandwidth, who is using them, when they are being used, what activities are prevalent.
- Have visibility and control of the network.
- Gather data for network and capacity planning.
- Troubleshoot issues.
- Implement network forensics.

The rest of the paper is organized as follows. Section two describes the NetFlow-Lite architecture and flow format. Section three covers NetFlow-Lite implementation both on the switch and collector side. Section four describes how the implementation has been validated against real traffic. Finally open issues and future work are described on section five.

II. NETFLOW-LITE

In essence, the NetFlow-lite solution consists of three elements:

- The switches that supports NetFlow-lite functionality and churn out NetFlow-lite data.
- The converter that aggregates the data into format understandable by NetFlow collectors in today's market place
- The NetFlow collector that collects and analyzes not only information originated through NetFlow-lite, but also NetFlow data gathered from different parts of the network, all through standard IPFIX format (or NetFlow version 9).

The converter implements the flow cache by populating it using the sample packets stored on the received flows, and not doing a simple 1:1 flow format conversion. It then exports the flows in standard NetFlow V5/V9/IPFIX to a standard NetFlow collector. In a nutshell, the NetFlow-Lite converter acts as a flow collector with respect to the switch as it collects NetFlow-Lite flows, and as a probe for the flow collector.

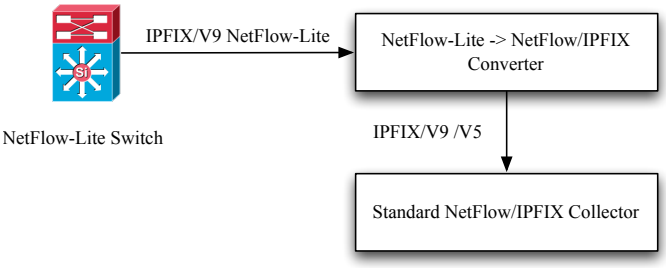

Figure 1. NetFlow-Lite Architecture

In order to preserve bandwidth usage for links on the path between the switches and the converter, an option is being provided to specify the number of bytes in the raw packet section that will be included in the export packet. In addition, it is preferable that the converter is located near the switch in order to avoid taking up extra forwarding bandwidth.

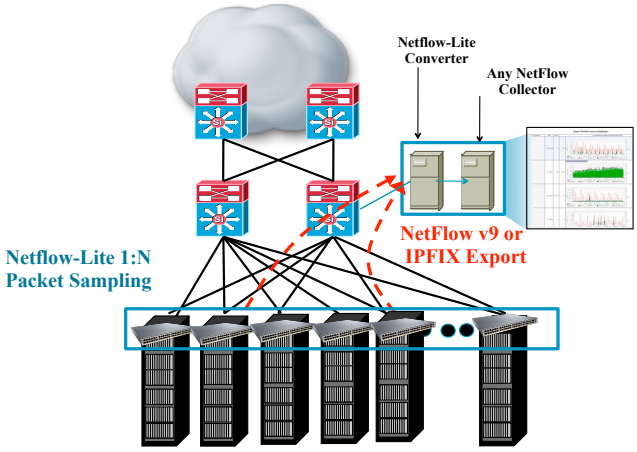

Figure 2. NetFlow-Lite Enabled Data Center Architecture

The figure above shows a NetFlow-lite enabled data center architecture, where NetFlow-lite samples incoming traffic on the TOR (top of rack) switches. The converter sits between NetFlow-lite capable switches and NetFlow collectors,

extracting the information from the raw packet section, such as IP address, TCP/UDP ports, etc. and aggregate them into a local flow cache. The flow cache can be exported to any existing NetFlow collector for analysis and correlating.

With larger data center, a zonal design is recommended. In that case, a converter is placed per "zone" to be responsible for aggregating and converting NetFlow-lite packets within the zone. Converters from different zones can be feeding the aggregated NetFlow data into a centralized NetFlow collector in order to achieve a data center-wide network visibility.

Flow Format

A switch with Netflow-lite functionality observes ingress traffic and sample packets at 1-in-N rate at the monitoring point, for example, an interface on the switch. The sampled packets are exported in standard NetFlow version 9 or IPFIX format. IPFIX and NetFlow version 9 differs from previous version in that it is template-based. Template allows the design of extensible record format.

It consists of:

- Template FlowSet: a collection of one or more template records that have been grouped together in an export packet.
- Template record used to define the format of subsequent data records that may be received in current or future export packets. It is important to note that a template record within an export packet does not necessarily indicate the format of data records within that same packet. A collector application must cache any template records received, and then parse any data records it encounters by locating the appropriate template record within the cache.
- Data FlowSet: a collection of one or more data records that have been grouped together in an export packet.
- Data record: it provides information about an IP flow that exists on the device that produced an export packet. Each group of data records (that is, each data FlowSet) references a previously transmitted template ID, which can be used to parse the data contained within the records.
- Options template: a special type of template record used to communicate the format of data related to the NetFlow process.
- Options data record: a special type of data record (based on an options template) with a reserved template ID that provides information about the NetFlow process itself.

One of the capabilities of this extensible design is to allow the export of raw packet sections in the Data Record, which facilitates the export of NetFlow-lite sampled packets.

NetFlow-Lite enabled switches exports three different templates that contain:

- Data template that describes the structure of sampled packet export by the switch.
- Options template that describes the structure of sampler configuration data.
- Options template that describes the structure of interface index mapping data.

The options template describing the sampler configuration essentially exports the structure of the following pieces of information:

- An identifier for a given sampler configuration.
- The type of packet sampling algorithm that is employed (currently 1-in-N packet sampling).
- The length of the packet section extracted from the input sampled packet.
- The offset in the input sampled packet from where the packet section is extracted.

Templates are exported by default every 30 minutes, and they can be packed into a single export packet for reducing the number of transmitted packets.

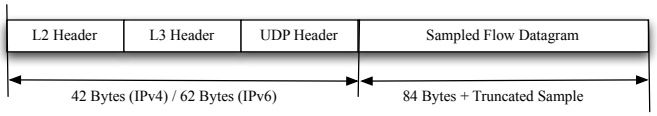

Figure 3. NetFlow-Lite Sampled Flow Datagram

From the flow format point of view, NetFlow-Lite flows are standard V9/IPFIX flows defined using a template. they contain packet section and other sampling parameters, but not the traditional fields such as source/destination IP address. In order to bridge between NetFlow-lite and NetFlow, and integrate NetFlow-lite into existing NetFlow solution, a converter is necessary in order to convert the information contained inside packet section, such as source/destination IP, TCP port, etc., into format understandable by the NetFlow collector on the market today.

NetFlow-Lite switches can adapt the sampling rate according to the switch port. This means that network managers can provide precise monitoring of selected switch ports by disabling sampling (i.e. 1-to-1 sampling rate), while using a higher sampling rate for all remaining ports. The use of the standard V9/IPFIX format prevents NetFlow-Lite converters to support a custom export protocol, while allowing them to be deployed anywhere in the network as long as they are reachable via IP. Another advantage is that future changes and extensions to the flow format, do not require changes on the collector as new fields can be accommodated into the exported flows simply my defining them into the exported template.

Flow conversion is transparent to existing NetFlow/IPFIX collectors and back-end tools. The use of sampling allows NetFlow-Lite to scale both in terms of number of ports and packets being monitored. Sampling rate can be adapted according to various parameters such as the total number of packets that are collected by a converter and also the number of switch exporters per converter.

III. IMPLEMENTATION

Due to its probe/converter architecture, supporting NetFlow-Lite has required both to enhance the switch and create the converter. No changes have been necessary on the collector side, as the converter emits standard flows in v5, v9 and IPFIX format.

Switch Implementation

On Cisco Catalyst 4948E switch, the sampling rate at which input packets are sampled is based on user configuration. The switch supports extremely (low) good sampling rate which allows for high quality of traffic monitoring. The sampling and export are both done in hardware, which does not put heavy load on control plane. Each sampled packet is exported as a separate NetFlow data record in NetFlow v9 or IPFIX format.

The switch implements a relatively inexpensive and not so stateful way of doing packet sampling and netflow export in hardware. The switch makes copies of the packets coming in and being forwarded through the switch, using appropriate rules in the classification engine that identify packets coming from monitored interfaces. The original packet undergoes normal forwarding and switching treatment through the device. The copies undergo a two-level sampling process.

At the first level, the copies of packets from various monitored interfaces are generated and sent to a transmit queue where a credit rate limiting scheme is applied. This credit rate mechanism is called DBL (Dynamic Buffer Limiting) and is proprietary to the Cisco Catalyst switches. DBL is used as an active queue management mechanism normally on the switch but in this case it is ingeniously being used for first level selection of sampled packets.

DBL credits are applied to a monitor and refreshed in a time based fashion that allows enqueue of packets to the transmit queue such that there are enough packets from a monitored interface to match the user configured sampling rate. Whenever a packet from a monitor is enqueued to the transmit queue, the credits for that monitor get decremented. The credit lookup is done through a hashing scheme that can take as input various packet fields and input port. This effectively provides the ability to sample packets as if on the input before packets from various monitors aggregate into the transmit queue.

The DBL credits and refresh frequency take into account the average packet size observed at a given monitor. Users may override the observed average packet size at a monitor and configure an average packet size for a monitor via CLI. The system will then use that average packet size in computing credits for traffic seen by that monitor.

Traffic flows from each monitor are isolated from traffic on other monitors because the DBL hash key masks are based only on the incoming interface or VLAN ID for port and vlan monitors respectively.

From the transmit queue the sampled packets are fed to a FPGA which does final sampling for packets from each monitor to eliminate extra samples. They are then exported in NetFlow version 9 or IPFIX format, assisted by the FPGA.

The combination of high sampling rate and user-configurable options provide a highly accurate sampling for NetFlow-lite. The hardware-assisted sampling and export offer a scalable solution with minimal impact to the control plane.

NetFlow-Lite Converter Implementation

The NetFlow-Lite converter has been implemented as an extension to nProbe [4], an open-source NetFlow/IPFIX probe/collector developed by one of the authors available for both Unix and Windows systems. As stated before, the flows emitted by the switch to the exporter are following the v9/ IPFIX guidelines thus from the flow format point of view no changes have been necessary. The main changes in nProbe have been:

- Ability to interpret the received NetFlow-lite flows.
- Extract the packet samples.
- Use samples to populate the flow cache.

In addition to packet samples, the flows emitted by the switch contain additional information that is necessary to properly support NetFlow-Lite, including:

- The sampler named and id (configured into the switch)that has sampled the packet.
- The sampling algorithm and size of the sampling pool, used by the sampler.
- The original packet length before cutting it to the specified snaplen.
- The packet offset of the received sample, as the switch can be configured to emit sampled packet starting from a specific offset (the default is 0) after the ethernet header.
- The switch interface on which the packet has been sampled.

Switch samplers are responsible to select packet to sample. A switch can define many samplers, and thus each switch port can potentially have a specific sampler. This allows for instance to have a per-port sampling rate, but it requires the converter to store this information as the received samples need to be scaled based on the sampler that has emitted them.

In order to enhance the exporter performance, it is possible to configure the switch to send flows to a pool of UDP ports and not to a single one. The switch sends the flow templates to the first port of the pool, and flow samples to the remaining port. Currently the destination ports are selected in round-robin in order to balance the load on the collector side.

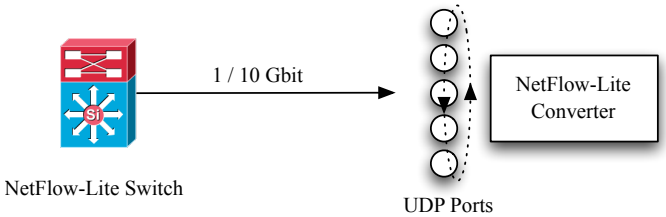

Figure 4. NetFlow-Lite Collection

This has been an important change as it has allowed the converter to boost its performance. In fact, NetFlow collectors usually are designed to handle a limited number of flows per second [14] that are often dumped to persistent storage after filtering and aggregation. In the case of the NetFlow-Lite converter the number of received flows can be very high and exceeds the rate of 1 million flows/sec, whereas a high-end NetFlow collector can very seldom handle sustain rate of a couple of hundred flows/sec. The number of collected flows can be quite high if the switch is configured with a 1:1 sampler on a high-traffic port. Unfortunately as all the templates are send to a single UDP port, it is not possible to spawn multiple independent converters, one per UDP port, so that they could each analyze a portion of the traffic. Furthermore as the switch is selecting destination ports in round robin, it can happen that two sampled packets belonging to the same flow are sent to different UDP ports. The use of 16 multiple collection ports has allowed nProbe to successfully collect and convert up ~500K flows/sec per switch with a single threaded instance. Unfortunately this performance has been enough and thus a different solution had to be developed.

Leveraging on the experience of the PF_RING project [15], in order to further boost converter performance, we decided to exploit multi-core computer architectures by developing a kernel module for expedite operations. The idea is to perform in-kernel NetFlow-Lite collection driven by the user-space nProbe converter.

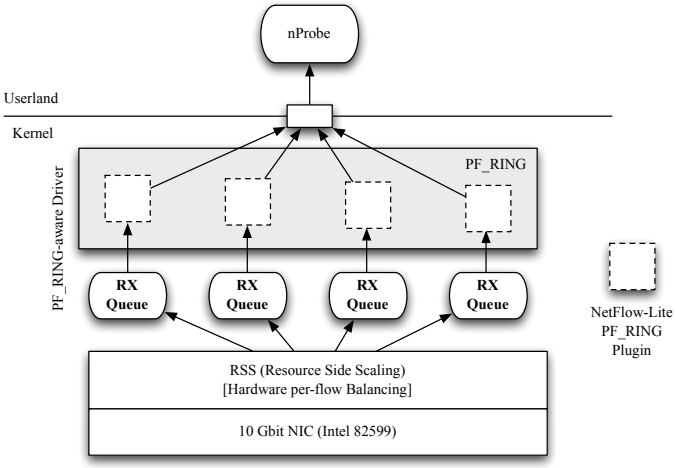

Figure 5. NetFlow-Lite PF_RING Plugin

nProbe sets a PF_RING kernel filter for the IPv4/v6 UDP ports on which flows will be received, that instructs PF_RING to divert such packets to the kernel plugin without letting them continue its journey to user-space. The PF_RING kernel plugin implements flow collection by maintaining information about the received templates in kernel memory. Sampled packets are extracted from flows and sent to nProbe via a PF_RING socket. Along with the packet header and timestamp, PF_RING adds some metadata such as sampling information and interface Id, that have been extracted from received flows. Modern multi-queue adapters such as Intel 82599 allow cards to be partitioned into several RX queues, one per processor core. PF_RING exploits this feature and capitalizes on it by allowing each queue to work independently, and poll packet concurrently one per core. By means of a PF_RING-aware driver that pushes packets to PF_RING without using Linux kernel queueing mechanisms, packets are copied from the NIC buffers directly to the NetFlow-Lite plugin. As there is a single plugin instance, kernel locking has been carefully avoided when possible, thus each queue extracts sampled packets without interference from other queues. The only lock present on the plugin is used when templates are received and need to be copied in memory. As this information is shared across all queues, it is necessary to use a lock in order to avoid that a poller is using a template while it is updated. Nevertheless as templates are received very seldom (by default every half an hour) we can assume that no locking happens. An advantage of this solution, beside the increased processing speed, is that every PF_RING-aware network application can use the converted packet samples to implement monitoring. For instance by means of libpcap-over-PF_RING, applications such as tcpdump and wireshark can analyze received packets as if they were captured from a network interface, this without being aware of having been received encapsulated in NetFlow-lite flows.

The use of an external server-based converter can be detected by a flow collector as flows are sent by nProbe and not by the switch. In order to make NetFlow-Lite totally transparent to applications, nProbe has implemented automatic packet spoofing based on the source IP:port on which sampled flows have been received. Thus converted flows are not sent with the IP address of the server on which nProbe runs, but with the original IP:port of the switch that has sent the NetFlow-Lite flows. This information is propagated by the PF_RING kernel module to nProbe as part of the metadata information associated with each packet.

Collector Implementation

The collector receives and stores the NetFlow-Lite datagrams from the converter. Data is massaged and formatted then made available to the reporting front end. The reports are in turn used to optimize network performance. As previously stated, no change has been necessary to support NetFlow-Lite on the collector side with respect to standard NetFlow collection.

IV. VALIDATION

In order test and validate the implementation of NetFlow-Lite, several tests have been performed both in lab and also on real networks.

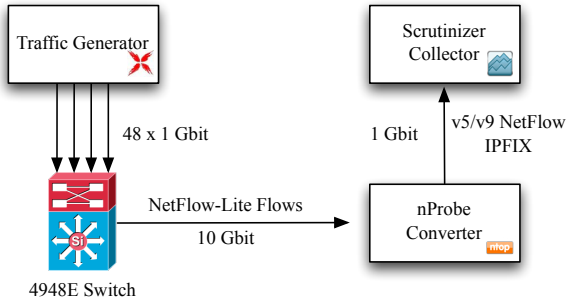

Figure 6. NetFlow-Lite Test Lab

In order to evaluate the switch implementation and the converter performance, a high-end IXIA traffic generator has flooded the switch sending traffic at wire-rate with minimum packet size on all 48 switch ports. The switch has been configured to send NetFlow-Lite flows to a 8-core Xeon server running various Linux versions including 64 bit Ubuntu 10.10 and RedHat ES6. On the server the nProbe 6.4.3 exporter was sitting on top of PF_RING 4.6.4 and the NetFlow-Lite kernel module. The switch has been connected to the converter on a 10 Gbit Intel 82599-based ethernet interface. A 10 Gbit interface has been used to both test the performance of the exporter when sending flows from multiple switches, and to flood the collector with flows. The Plixer Scrutinizer 8.5 flow collector has been installed on another server connected to the network with a 1 Gbit interface.

The test has confirmed that the sustained conversion rate sustained per nProbe converter instance has been 500K flows/sec when receiving flows over UDP, and 1M flows/sec using the PF_RING kernel module. Converted flows have been sent to Scrutinizer on various formats including NetFlow v5/v9 and IPFIX. Various test sessions have confirmed that collector users are unaware of the NetFlow-Lite to NetFlow/IPFIX conversion. Please note that on Windows platform nProbe also features NetFlow-Lite conversion but just over UDP.

A nice feature of the implementation on 4948E is the ability to specify different sampling rates based on switch ports. This is useful as network administrators can decide to disable sampling for those ports where there are critical services, and increase sampling rate on ports where no accurate monitoring is needed. In fact the use of sampling prevents nProbe from being able to report application protocol information including application and network delay (computed on the 3-way-handshake packets), and HTTP/VoIP traffic monitoring.

V. OPEN ISSUES AND FUTURE WORK

Although the converter performance is enough for many users, a future work activity is definitively related to how to improve this conversion. Currently the switch sends flow to all configured UDP ports in round-robin. The ethernet interface hashes flow packets using RSS [16], thus distributing them based on the destination UDP port and not based on the sampled packet contained in the received flow. This is not ideal as in order to keep the NetFlow cache consistent, it is not possible to enhance the converter performance by spawning one nProbe instance per RX-queue. This is because RSS does not guarantees that packet samples belonging to the same flow will be sent to the same RX queue.

In order to address this issue that limits the converter performance, we are currently enhancing the PF_RING NetFlow-Lite plugin so that received samples will be re-hashed based on the sampled packet and not on RSS. This will allow one nProbe instance per RX queue to be spawn thus maximizing performance. Please note that the kernel plugin keeps track of received templates and thus guarantees flow conversion consistency also across multiple switches all sending flows to the same converter server. This performance enhancement is also compatible whenever configured switch samplers have a packet offset greater than 0 (i.e. when the offset is zero the sampled packet contains the whole ethernet header) but not larger than 14 bytes (i.e. the length of the ethernet header). This is because the plugin does not hash samples based on the ethernet header but rather on the IP header that is also used as flow key inside the converter cache.

VI. FINAL REMARKS

This paper has described the design and implementation of NetFlow-Lite. By means of it, network administrator can provide network visibility similar to NetFlow/IPFIX while maintaining switching performance. The validation phase has confirmed that the use of a NetFlow-Lite to NetFlow/IPFIX converter is seamless for the end-user of the flow collector and that the converter performance is high enough to allow network administrators to reduce sampling (if any) on switch ports where critical services are running. The flexibility of NetFlow-Lite combined with the lack of changes on the collector side, smooth its adoption and makes it a good candidate for providing visibility on switched environments.

VII. ACKNOWLEDGMENTS

The authors would like to thank the NetFlow-Lite team and in particular Manikandan Arumugam for his help and support throughout the project and testing phase.

[1] B. Claise, Cisco Systems NetFlow Services Export Version 9, RFC 3954, October 2004.

[2] B. Claise, Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information, RFC 5101, January 2008.

[3] J. Case et al, A Simple Network Management Protocol (SNMP), RFC 1157, 1990.

[4] L. Deri, nProbe: an Open Source NetFlow Probe for Gigabit Networks, Proc. of Terena Network Conference, 2003.

[5] P. Lucente, pmacct: Steps Forward Interface Counters, Technical Report, 2008.

[6] T. Zseby and others, Sampling and Filtering Techniques for IP Packet Selection, RFC 5475, March 2009.

[7] N. Duffield, Flow Sampling Under Hard Resource Constraints, Proc. of SIGMETRICS '04, 2004.

[8] B. Choi and S. Bhattacharyya, On the Accuracy and Overhead of Cisco Sampled NetFlow, Proc. of ACM SIGMETRICS '05, 2005.

[9] R. Sommer and A. Feldman. NetFlow: Information Loss or Win, Proc. of ACM SIGCOMM Internet Measurement Workshop, 2002.

[10] J. Clearly et al., Design Principles for Accurate Passive Measurement, Proc. of PAM Conference, 2000.

[11] J. Jedwab et al., Traffic Estimation for the Largest Sources on a Network Using Packet Sampling with Limited Storage, HP Labs, 1992.

[12] K. McCloghrie and M. Rose, Management Information Base for Network Management of TCP/IP-based internets: MIB-II, RFC 1213, 1991.

[13] Cisco Systems, Configuring NetFlow-Lite Software Configuration Guide, Release 15.0, May 2011.

[14] Y. Fragiadakis et al., User and Test Report of the NetFlow Collector, Geant II Project, 2009.

[15] L. Deri, Improving Passive Packet Capture: Beyond Device Polling, Proc. of SANE '04, 2004.

[16] Microsoft, Scalable Networking: Eliminating the Receive Processing Bottleneck — Introducing RSS, WinHEC (Windows Hardware Engineering Conference) 2004.