Flow measurements from the packet-switched NREN PIONIER: technology and experience

.

Wiktor Procyk, Szymon Trocha
Poznan Supercomputing and Networking Center
Noskowskiego 12/14, 61704 Poznan, Poland
{wiktor.procyk, szymon.trocha} @man.poznan.pl

KEYWORDS:
sFlow, network management, rrd

ABSTRACT

In order to satisfy all the requirements computer networks have to be monitored. Most modern network devices support sFlow – a source of very detailed information about traffic in the network, but only few, mostly commercial, applications are able to use it. We created a framework for flow measurement based on sFlow and RoundRobinDatabase, so information can be easily accessed by NOC and network administrators.

I. INTRODUCTION

During the recent years computer networks have dramatically increased the core bandwidth. Widespread Internet becomes more and more popular. All these factors are conductive to the deployment of new services like grids, videoconferencing, etc. But the bandwidth is not everything that the today's applications require. The quality of service – latency, jitter, packet loss - becomes equally important. Computer networks have to be monitored and managed at every layer of ISO/OSI model to satisfy all the requirements and expectations and to provide service consistent with an appropriate service level agreement (SLA) .

There are many multi-vendor network management applications and measurement tools available on the market and used by network administrators [1]. In practice, many National Research and Education Networks replace them with GNU GPL-based software like MRTG [2] or Nagios [3] because of lower price and more flexibility.

The trouble is that those tools still have poor support for network layer.

The recent years of network operation have shown that the monitoring of the network layer is a must, especially in case of new services like digital channels. If there is quality degradation in the network, it has to be determined which part of the network it is responsible for. Detailed traffic measurements are necessary to efficiently engineer the network.

A powerful source of information about network is sFlow [4] - multi-vendor sampling technology embedded within switches and routers. It provides the ability to continuously monitor application level traffic flows at wire speed on all interfaces simultaneously.

The current tools for network monitoring are limited in their support of sFlow. The need to automate the analysis and presentation is critical for efficient operation of wide scale network NOC like NREN NOC. This problem of providing NOCs with data collection and presentation tool is an area of our research focus.

In order to have a better insight into network traffic we decided to use the application sflowtool [5] which listens for sFlow samples, decodes them and displays as a text in a human readable format. A Perl script is used for storing the traffic data in the RoundRobinDatabase [6] format common for GNU GPL licensed network management tools.

Because of hard interpretation of the raw data obtained using the aforementioned application we decided to create a framework alternative to the original approach.

The first attempt at implementing this approach was undertaken at the Polish NREN PIONIER [7] NOC for generating Ethernet VLAN traffic reports. The system which is available via Web interface gave the NOC the capability to quickly aggregate, analyse and present data. It can be easily extended to provide other information about the type of service, ports, AS numbers available with the use of sFlow.

In this paper we illustrate technology for a monitoring framework for collection and presentation of sFlow data. We also present the deployment experience gained with using it within the Polish NREN and highlight the impact of this application on the NOC operation. The results of the prototype implementation will also be presented in a form of network traffic measurements, followed by a discussion on future work.

II. PROBLEM

The first attempt to monitor network traffic using sFlow samples was undertaken using Perl script sflowRRDLoad available as one of sflowutils. The aim of this script is to monitor the usage of the interfaces.

The results were not satisfying. The sFlow sampling rate on the PIONIER network elements is set for all interfaces, so the number of sampled packets from a

given interface depends not only on the sampling rate, but also on the sFlow status on other interfaces (enabled/disabled). Each change of sFlow configuration caused changes in the outcome of the monitoring system making the results difficult to interpret and not reliable. We will discuss it on an example. Let us assume that we have eight interfaces (numbered from 1 to 8), each one with the same volume of traffic. sFlow is enabled only on the first interface, but the sampling rate is set per the whole device. So we count every packet crossing the switch but the sampled packets are taken only from interface number one, which is sFlow-enabled. And now let us analyse the following cases:

- traffic on interfaces 5 to 8 is tripled, so the total volume traffic crossing the switch is doubled - we receive twice more samples from the first interface.

- sFlow is also enabled on the second interface - the total number of samples remains the same, but only half of them concerns traffic on the first interface.

However, this is only part of the problem. Cases described above do not apply to devices with sFlow sampling set per each interface independently. In such a case, if we used the aforementioned sflowRRDLoad Perl script, it should be enough to multiply interface sampled traffic volume by sampling rate to get the total volume of traffic for this particular interface. It may also happen that the results differ from the ones obtained using the MRTG tool and SNMP counters. One of the possible reasons is that there can be packet loss between the sFlow-enabled device and the sFlow collector. The question then arises whether we should tune the multiplying factor in order to achieve matching results.

Another problem to overcome is data loss for VLANs with very low network usage. Every sampled packet feeds the appropriate period in the RRD file. For periods where no packet has been sampled, the RRD tool does not write data point to the database. The problem occurs during time aggregation when none of the techniques used by the RRD tool engine is acceptable, e.g. if we want to aggregate periods of time with and without data points the RRD tool can only take one of two actions: mark the whole period as "no data available" or calculate the average for a given volume of traffic values and extend it to cover the whole time period.

III. MONITORING FRAMEWORK

We first define some notation we use in this section:

- T the total traffic on the interface;
- v the traffic on a particular VLAN on that interface;
- s VLAN's share on that interface.

Using the sFlow and Perl script has many advantages. sFlow can deliver very detailed information about network traffic and Perl is flexible in processing data – it can be easily tailored to satisfy all the needs.

Originally for each sampled packet the data has been acquired and then appropriate RRD file updated with the value corresponding to the sampled packet size. Then the RRD system takes care of putting the value in an appropriate period. But, as we mentioned, the number of sampled packets may vary regardless of the real traffic volume.

We also observed that if traffic did not change, the proportions between different VLANs remained at the same level, even if we changed sampling rate or sFlow status on other interfaces.

Because of difficulties in the interpretation of absolute values, we decided to monitor relative values. For every VLAN attributed to an interface we monitor its share in the total traffic T on that interface which is $s = v / T$.

With the solution proposed here to observe the VLAN traffic, we have several stages of processing:

Step 1 – acquiring data

From every sample we take the IP address of agent, input and output interface, VLAN number and packet size.

Step 2 – dividing time into periods

The current timestamp is taken from the localhost and checked if the measurement period has elapsed.

Step 3 – updating files

If the measurement period has elapsed, we update files storing data collected so far in step 4 (ratio s). For each file we check the last timestamp update, and if it is older than the end of the previous period, we feed all missing periods with value 0. Data is stored in files with the following naming convention: <agentIP>-<interface>-<VLANnumber>.rrd. After updates all the collected data are flushed and a new period begins.

Step 4 – data processing

Data acquired in step 1 is collected. It is stored in two hash tables: one for interfaces and another one for VLANs on the interface.

In comparison to the original approach, only step 1 remains untouched. Because we calculate the share s, we have to wait for the time period to finish (step 2) and if it has not elapsed, store the value temporarily in the hash table (step 4). We increment the traffic volume T and VLAN traffic v with the size of the sampled packet. Step 3 is the clue. First of all, for every VLAN we calculate the ratio s. It is worth remembering not to divide by zero in case of unidirectional traffic.

Before we explain how the framework solves the problem of filling RRD files with the 0 value, let us focus on the RRD options. There are two important configuration options: xff and heartbeat. Xff concerns aggregation when data from a few short periods of time is aggregated into one value to save the occupied filesystem space. If we have periods of time with some values mixed with periods when no data was available, xff is a threshold how many periods with out data are taken into account during aggregation. For example, xff=1/3 means that to aggregate 6 periods into 1 it is required to have at least 4 values; otherwise the whole period is marked as "no data".

And now if we do not have any sampled packets, we will not update the RRD file. Let us take a look at what can happen then:

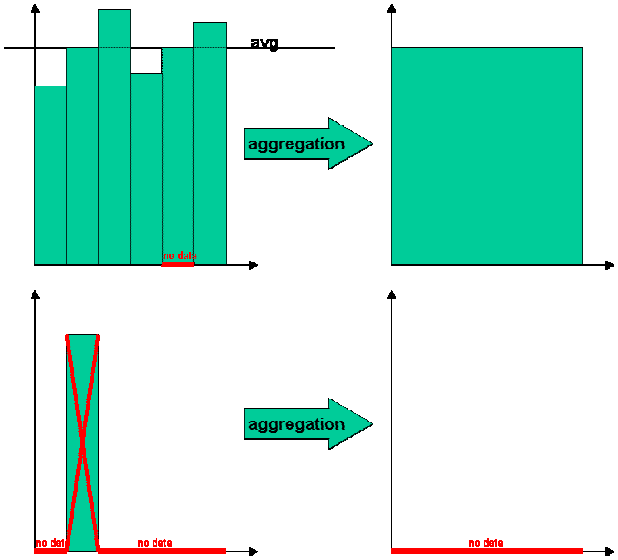

Figure 1. Two cases of aggregation mixed periods with values and period with “no data”

In the first case shown on Figure 1 one period is marked “no data”, so that period is excluded from calculating the average and as a result this period is treated as it would have an average value.

In the second case there is not enough values, so these values are lost and whole aggregated period is marked as “no data”.

Our experience shows that we can avoid such situation using the heartbeat option. This option tells us how long the last update remains valid. For example, if we have 5-minute periods, heartbeat is set for 24h we can feed RRD twice a day and there will be no “no data” periods because the last update value will be split to fill all the periods between updates. So generally very long heartbeat solves our problem. Unfortunately, the tests run over the national network showed that it leads to distortions and the total sum of traffic on all VLANs on the interface varies between about 95% and 105% of total interface traffic volume T, which can be confusing and not reliable.

To avoid distortions and keep the sum on the 100% level at all times before each RRD update we check the last update timestamp and fill all lacking periods with value 0.

If we want to get absolute values about traffic (in bps or Bps), it is enough to multiply VLANs share s by values taken from MRTG statistics (which use SNMP).

The solution presented above is a must in case of devices where sFlow sampling is set per the whole device. The main advantage remains the fact that we can change the sampling rate and the monitoring system does not require any changes nor reconfiguration. We also have no dilemma which results (MRTG or sFlow based) are closer to the real values.

To give the administrator a powerful tool to visualize the data, we also created a website frontend. It is written in PHP and uses the MySQL database. It consists of two tables: one for network devices and one for interfaces. The database is used to make a relation between rrd files produced by the MRTG and VLAN monitoring system.

IV. Implementation

The framework has been successfully implemented and is currently running in the Polish NREN PIONIER Network Operation Center for the generation of Ethernet VLAN traffic reports and substantially helps network administrators to diagnose packet-switched network traffic problems. The tool collects data from 16 BlackDiamond switches which operate the national backbone of 10 Gb/s and run 278 VLANs. There is a central data collector where the whole data is sent. The sFlow collector is running on the PIII 500 system with 512 MB RAM powered by FreeBSD. The CPU usage is about 5%, sflowtool and Perl script uses about 10MB of RAM. There are 5424 rrd files (including 2349 updated during the last 24 hours) occupying 350 MB disk space in total.

Figure.2 shows the Web interface for sFlow monitoring application.

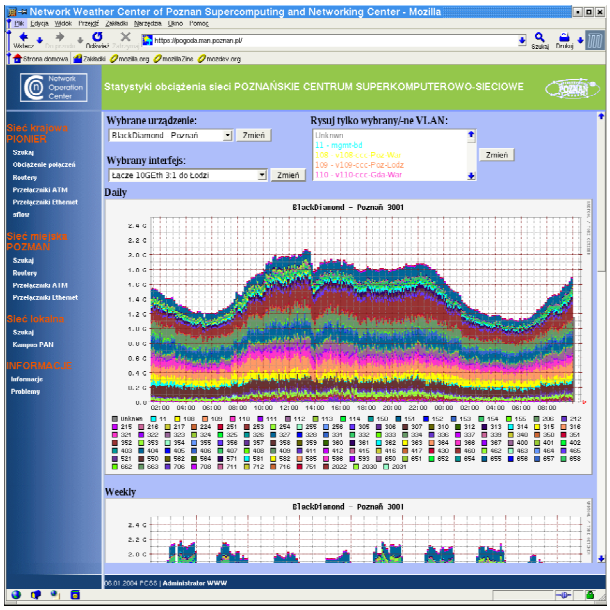

Figure 2 Website frontend for VLAN monitoring system

V. Conclusions and Further Work

The described framework uses the sFlow technology which is present in today’s widely deployed packet-switched networks and enhances tools which could be easily used to show traffic data and detect network problems in a user-friendly graphical way.

There are also some unresolved problems which remain as future work. We would like to reduce space occupied by files updated more than 1 month before. Growing popularity of the IPv6 protocol and the fact that it is supported by sFlow makes us consider the tool useful also for monitoring the IPv6 traffic

If we wanted to extend monitoring with other parameters, we would have to change the Perl script which requires deep analysis of the script itself and some Perl fluency. There is also a possibility of introducing some errors every time we modify the script and to crash the monitoring system. Works done in the area of Data Stream Management Systems [8] showed that for the future it is recommended to replace the Perl script by DSMS.

It is also worth analysing the possibility of archiving sFlow streams and to create an advanced filtering framework to make backward statistics possible for all possible network parameters.

ACKNOWLEDGMENT

The authors thank Tomasz Szewczyk of Poznan Supercomputing and Networking Center for his time and effort in implementing the framework.

REFERENCES

[1] M.Murray and K. Claffy, “Measuring the immeasurable: global Internet measurement infrastructure”, presented at PAM2001.
[2] The Multi Router Traffic Grapher (MRTG) system official homepage [Online]. Available: http://people.ee.ethz.ch/~oetiker/webtools/mrtg/
[3] The official Nagios website [Online]. Available: http://www.nagios.org/
[4] The authoritative website with information on sFlow, specifications, latest developments and products, that support sFlow [Online]. Available: http://www.sflow.org
[5] InMon’s scripts and utilities for analyzing sFlow data [Online]. Available: http://www.inmon.com/technology/sflowTools.php
[6] The official RRD website [Online]. Available: http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/
[7] Polish Optical Internet PIONIER [Online]. Available: http://www.pionier.gov.pl/eindex.html
[8] T.Plagemann, V.Goebel, A.Bergamini, G.Tolu, G.Urvoy-Keller, E.Biersack, “Using Data Stream Management Systems for Traffic Analysis – A Case Study”, presented at PAM2004 [Online]. Available: http://www.pam2004.org/papers/113.pdf

VITAE

WIKTOR PROCYK received the M.Sc. degree in computer science from Poznan University of Technology in 2000. He works in the Management Unit in PSNC. His interests include active and passive measurements and IPv6.

SZYMON TROCHA received the M.Sc. degree in computer science from Poznan University of Technology in 1998. He is Head of the Management Unit in PSNC. He is mainly involved in the network management applications planning and implementing.