An Open Source Environment for Capturing IP Traffic

Maciej Słomczynski
Poznan University of Technology
Poznan, Poland
Email: maciej@slomczynski.pl

Mariusz Głąbowski
Poznan University of Technology
Poznan, Poland
Email: mariusz.glabowski@put.poznan.pl

Maciej Sobieraj
Poznan University of Technology
Poznan, Poland
Email: maciej.sobieraj@put.poznan.pl

***Abstract*—This article presents a developed environment to enable collecting data providing detailed information about traffic in networks based on the IP protocol (Internet Protocol). Real traffic generated by individual users in a network within the domain of an Internet service provider is examined. The developed environment makes it possible to obtain information at the flow level and at the packet level and provides a solution using free software.**

I. Introduction

We are currently witnessing not only an upsurge of services based on the IP protocol, but also the further expansion of communication between not only humans but also between devices (Internet of Things) [1]–[4]. These phenomena have lead to significant changes in the characteristics of traffic generated by traffic sources (people, things) that should be taken into consideration as early as the designing and dimensioning stage in the network development, primarily in advanced IP access networks settings. A development of specific, efficient and exact methods for dimensioning of these networks requires first of all accurate insight into the character of transmitted traffic. Information about the parameters of traffic generated by users, as well as information about the quality parameters (quality of service, quality of experience) that are expected by them, make appropriate mapping of the resources demanded at the packet level into the resources demanded at the session (call) level possible, and, in consequence, provide an opportunity to work out appropriate methods for network dimensioning using the approach based on the idea of the so-called equivalent bandwidth [5], [6].

In order to understand fully the character of traffic generated in modern IP access networks, thus making it possible to determine values of equivalent resources in the future, the article presents the developed environment that allows researchers to obtain extensive information about the structure of transmitted traffic streams.

The remaining sections of the article are organised as follows. Section II presents the basic assumptions and the architecture of the controlled environment that captures IP traffic. Section III discusses possibilities for using the *NetFlow* protocol to retrieve information about packet flows of transmitted traffic [7]. This section also proposes a software and procedures suitable for the purpose that make capturing, recording and storage of information obtained using *NetFlow* possible. It also provides the reader with an evaluation of the amount of captured data depending on their range of extraction. Section IV concludes the paper.

II. Assumptions and architecture of the environment for capturing IP traffic

In the initial stage of the development of the environment for collecting information about traffic generated in IP access networks the assumption was made that only traffic generated by individual end users would be analysed. This traffic will be thus involved in all types of service available on the Internet such as: WWW, e-mail, VoIP, Video, etc.. Traffic generated by system servers that service other Internet users will be excluded from the study, as well as traffic generated by applications monitoring/managing the network operation and data archiving systems.

The initial assumption in the study was that in the developed environment there were two levels of measurement (capturing/retrieving information). The first deals with collecting information about streams from an edge router and the backbone network (session level), the other being restricted to collecting information at the packet level as a result of capturing all packets and tagged information within individual sessions (packet level).

A general diagram of the network under investigation is shown in Figure 1. The main element of the network, from the point of view of measurements to be performed, is a server with Linux operating system, *Debian Squeeze* distribution, that acts as an edge router (denoted in Figure 1 as the Linux–router). This server is also used as a router BGP [8], bandwidth limiting for some users and translating addresses (NAT – Network Address Translation) for most of users. In addition, for the sole purpose of the study, the router can also perform the function of the so-called Flow exporter, i.e. a device that collects and transmits information about particular flows (sessions). A dedicated server is introduced to the proposed environment, denoted in Figure 1 as the *Linux-collector*, which collecting data generated by "flow exporter". In order to relieve the edge router from managing the function of the "flow exporter", the *Linux-collector* sever can perform both the managing function designed to capture information about transmitted flows and the function of a device that collects information of captured flows.

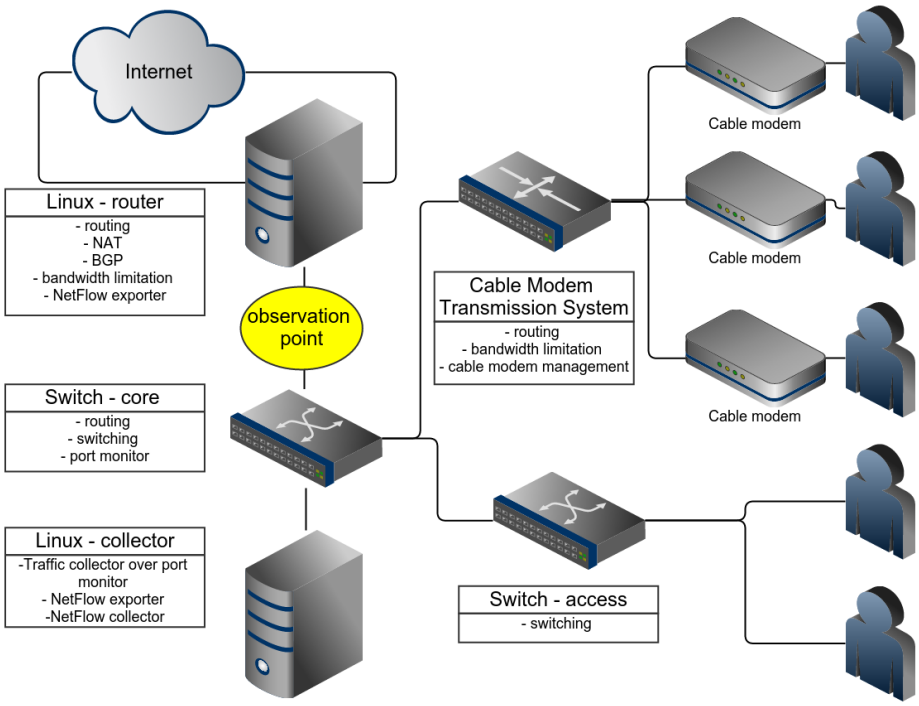

Fig. 1. Diagram of the network under investigation

The developed system for retrieving data offers a possibility to analyse network traffic to the level of a single packet, as well as assign each of transmitted packets to the flow generated by individual users of the network. Thus obtained data, following a long-term and periodic analyses, will allow the researcher to generate statistical flow patterns generated by demands of particular services. Patterns worked out in the process will include information on the parameters of the access link through which they have been introduced into the network.

Future studies will make it possible to go deeper into the structure of real traffic in the network in terms of its assignment to individual types of services. With future investigations that will be performed on periodic basis, a possibility will arise to monitor and observe trends and changes in bandwidth allocations for individual services that occur within time. Monitoring of trends in changes in traffic distribution and of the entirety of the bandwidth will make it possible to predict and anticipate bandwidth demand in Internet access networks (bandwidth demand forecast).

III. Tools for IP traffic capturing

A. Data stream information capturing at the session level

To retrieve information about transmitted data flows (execution of the function "Flow exporter") the *NetFlow* protocol (in its 5 pro version) is proposed [7]. This protocol is implemented in most network solutions. In Figure 2 shown *NetFlow* generation scheme. It should be noted, however, that the implementation of this protocol in devices manufactured

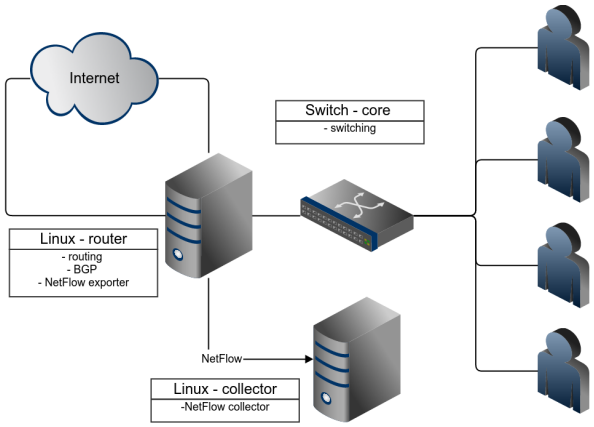

Fig. 2. Diagram showing *NetFlow* data generation

by a number of producers (e.g. Cisco) generate only sampling data in order to decrease the load of these devices. Depending on a version of the operating system employed in network devices, there is often a possibility to introduce changes in the frequency of sampling.

However, using sampling data collected only within a predefined time interval results in a deliberate exclusion of information about a great number of traffic streams. In order to capture information about all flows, an application of the *fprobe* program, in its most common version, i.e. 1.1-7.2, was proposed in the initial stage of the study. The analysis of data retrieved as a result of the application of this program showed,

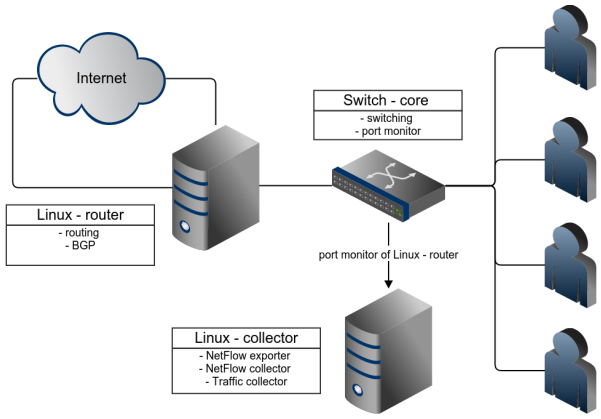

Fig. 3. Network measurement diagram

however, that it generated a constant value of time in the field "SysUptime", included the *NetFlow* message header [7]. This, in consequence, made it impossible to unequivocally place the transmitted stream on the time axis. The same shortcoming in the distribution evaluation also occurred in the latest fprobe program, i.e. 1.1-7.3.

A successful solution to the problem was finally found in the application of the software packet *softflowd* in its 0.9.9 version (the 0.9.8 version proved to have been unstable).

Data retrieved as a result of the application of the *softflowd* program (flow exporter functions) include information about the source and target IP addresses, as well as specify source and target numbers of ports used for each of the transmitted stream. It should be noted, however, that the retrieval of such detailed data by softflowd generates heavy load for the server processor that performs its function as an edge router. Considering the critical functions assigned to the edge router important in the overall operability of the operator, such as translation of addresses and limiting bit rate for individual users, it is impractical to allow the load of the sever processor to increase up to 100%. This would effect in a drastic reduction of stability of the network. To maintain full operability of a monitored network, the tasks related to user data transmission, as well as those related to the retrieval of information about these data, were divided between particular processor cores in the "Linux-router" edge router.

The Flow-capture program from the package *Flow-tools* [9] was used to perform the functions related to collecting data retrieved by *NetFlow*, i.e. the *Flow collector* function. To decrease the load of the main server in the network that performed the function of the edge router, the *Flow collector* function was activated and ran on a server specially dedicated for the purpose. This server can also preform the function of the flow exporter and of a device that can capture the entirety of traffic (the measurement level is discussed in the following section of the article). Figure 3 shows developed data collection scheme.

It should not be forgotten though that in the considered

TABLE I
COMPARISON OF DATABASE ENGINES IN IMPORTING *NetFlow* DATA

	NetFlow data volume [MB]	number of flows (records)
NetFlow probe	10.4	837623
database engine	data base volume [MB]	time loading to database [s]
MyISMA	84.9	143.81
innoDB	127.6	268.96

architecture all *NetFlow* data, as well as all headers of analysed packets, come from one source. In the case of the instance of a number of different *NetFlow* sources and packet headers, an important element in the cumulative analysis would be to synchronise clocks in such systems to maintain coherence of retrieved data.

B. Storage of information about captured data streams

In order to ensure the possibility of storing large amount of information about captured streams and of making reliable access to them easy, a decision was made to use MySQL database for storing captured data. Two database engines most frequently used in MySQL were considered: MyISMA and innoDB. Despite certain theoretical prerogatives that indicated innoDB as the most appropriate engine, it was the MyISMA engine that turned out to be faster in importing data into MySQL by 47% and, from the point of view of the ultimate capacity of the database, was more efficient by 33%. Results are shown in Table I.

The *Linux–collector* server used for the purpose of the study was equipped with two four-core 2.50GHz. processors. The server was also equipped with SSD (solid-state drive) disks combined in RAID 0 (Redundant Array of Independent Disks) using hardware matrix controller. This resulted in virtually no delays in the system caused by the waiting time for an input/output operation during the data load into the database. In practice, this means that the critical point was a single core of the processor used by a specific process of MySQL. To accelerate the input process of the total *NetFlow* data into the database, parallel data load into the database using a number of scripts was used. This did not accelerate the process of data recording of a single process, but made the cumulative throughput to be significantly increased. Every data recording process by the MySQL database imposed load on one processor core at about 100%. At the same time, the flow-export program itself (from the package flow-tools) that serviced *NetFlow* data recording to MySQL, occupied about 15% of another core.

To identify and determine necessary resources, a two-day pilot test was conducted. Table II shows the amount of data with which we are dealing. It is scaled in the amount of data generated for each day for each 100 Mb transmitted traffic

C. Capturing of information about data streams at the packet level

To capture data at the packet level, the measurement scheme prepared prior to the study and presented in Fig. 3 was used. The main elements of this scheme are the *Linux-collector*

TABLE II
A JUXTAPOSITION OF THE AMOUNT OF DATA GENERATED BY *NetFlow*

description	flow	database disk space	*NetFlow* disk space	loading to database time
unit	flows/day/100Mb	GB/day/100Mb	GB/day/100Mb	hour/day/100Mb
value	171555292	16.96	2.0	7.88

TABLE III
VOLUME OF DATA OBTAINED BY TCPDUMP

Parameter	Volume	Unit
Average amount of data – headers only	105898	kB/min/100Mb
Average amount of date – whole packets	879588	kB/min/100Mb

dedicated server and the port monitor function activated on the main core switch of the operator. The *port monitor* function is based on a duplication of incoming and outcoming traffic from the port of the switch under scrutiny. Traffic is directed to an indicated port to which a monitoring device is being connected. The port monitor function did not generate any observable load for the switch.

Tcpdump [10] was chosen to capture data. *Tcpdump* is a standard package in most Linux distributions. Capturing all information involves generation of a large number of data. This caused problems with capturing, storage and further analysis of the data. Due to the sheer amount of information and the confidentiality of data, the content transmitted by layer four protocols in the OSI model (Open Systems Interconnection) was decided to be rejected, while this included TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) protocols.

The proposed *tcpdump* tool makes it possible to get a packet capture in its demanded length only. To define an appropriate length for packets that are necessary for capturing it was necessary then to determine the maximum available length used for the packet header. A header of the second layer in the OSI model has the length of 14B. The IP header can have up till 24B. The UDP header has a fixed length of 8B and is always shorter than the TCP header. The first four bits of the twelfth octet of the TCP header define its length. The value of the first four bits of the twelfth octet multiplied by four makes the length of the TCP header expressed in bytes. Thus, the maximum length of the TCP header is 60B. Summing up the above calculations - in order not to omit any crucial data it was necessary to capture the first 96B of each of the TCP packet and the first 46B of every UDP packet within the second layer of the OSI model.

By the a two-day pilot test all transmitted traffic was captured as well as traffic in packets truncated after the first 96B. As shown in Table III after the truncation of packets, there were 12% remaining data relative to the total traffic.

In the study, another method for capturing TCP, UDP and remaining traffic was also developed. As a result, it was possible to capture only the first 46B from the UDP header. As shown in Table IV this effected in a decrease in the volume (size) of captured data to 7% as compared to the total traffic.

TABLE IV
VOLUME OF DATA OBTAINED BY TCPDUMP WITH CONSTRAINED UDP PACKET CAPTURING

Parameter	Volume	Unit
Average amount of data – TCP headers	53066	kB/min/100Mb
Average amount of data – UDP headers	4982	kB/min/100Mb
Average amount of data – other headers	371	kB/min/100Mb
Average amount of data – all headers	58420	kB/min/100Mb
Average amount of data – whole packets	752885	kB/min/100Mb

The samples from Table III and IV were taken in some other period, which resulted in differences in the average numbers of data for 100Mb.

IV. CONCLUSION

This article presents the architecture, assumptions and core elements of an environment for collecting data on the properties of traffic transmitted in present-day IP networks. The application of the *NetFlow* protocol makes it possible to obtain a solution in which the demand for resources, both disk resources and computational resources, is small enough to be used in real operator's networks without any detrimental influence on their efficiency.

The information about the traffic structure at the individual packet level and the individual stream level obtained as a result of further measurements will allow the average size of resources demanded by traffic streams of identifiable class of services to be determined. This, in turn, will make it possible to develop simple engineering methods (that make use of the concept of the equivalent bandwidth) for dimensioning the multiservice IP network.

ACKNOWLEDGMENT

The presented work has been funded by the Polish Ministry of Science and Higher Education within the status activity task "Struktura, analiza i projektowanie nowoczesnych systemów komutacyjnych i sieci telekomunikacyjnych" in 2015.

REFERENCES

[1] M. Grajzer, T. Zernicki, and M. Głąbowski, "Nd++ – an extended IPv6 neighbor discovery protocol for enhanced stateless address autoconfiguration in MANETs," *International Journal of Communication Systems*, vol. 27, no. 10, pp. 2269–2288, 2014. [Online]. Available: http://dx.doi.org/10.1002/dac.2472

[2] M. Grajzer and M. Głąbowski, "Performance evaluation of neighbor discovery++ protocol for the provisioning of self-configuration services in ipv6 mobile ad hoc networks," in *Telecommunications Network Strategy and Planning Symposium (Networks), 2014 16th International*, Sept 2014, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6959266

[3] D. Lake, A. Rayes, and M. Morrow, "The Internet of Things," in *The Internet Protocol Journal*, vol. 15, no. 3. Chief Technology Office, Cisco Systems, Inc., September 2012.

[4] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[5] A. Pras, L. Nieuwenhuis, R. M. van de, and M. Mandjes, "Dimensioning network links: A new look at equivalent bandwidth," *IEEE Network*, vol. 23, no. 2, pp. 5–10, March 2009. [Online]. Available: http://doc.utwente.nl/65443/

[6] N. L. S. Fonseca, G. S. Mayor, and C. A. V. Neto, "On the equivalent bandwidth of self-similar sources," *ACM Trans. Model. Comput. Simul.*, vol. 10, no. 2, pp. 104–124, 2000.

[7] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954 (Informational), Internet Engineering Task Force, Oct. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3954.txt

[8] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271 (Draft Standard), Internet Engineering Task Force, Jan. 2006, updated by RFCs 6286, 6608, 6793. [Online]. Available: http://www.ietf.org/rfc/rfc4271.txt

[9] "flow-tools - Open source program to collect, send, process, and generate reports from NetFlow data." [Online]. Available: https://code.google.com/p/flow-tools/

[10] "TCPDUMP - Open source command line packet analyzer." [Online]. Available: http://www.tcpdump.org/