

ABW - Short-timescale passive bandwidth monitoring

Sven Ubik (CESNET, Czech Republic), Demetres Antoniadis (ICS-FORTH, Greece),
Arne Oslebo (UNINETT, Norway)

Abstract

Bandwidth usage monitoring is important for network troubleshooting and planning. Traditionally, used bandwidth is computed from router interface byte counters read by SNMP. This method only allows to check long-term averages of the total used bandwidth without information about short-term dynamics and without knowledge of what applications are consuming most bandwidth.

We describe the architecture of a novel passive bandwidth usage monitoring application. This application uses packet capture and advanced processing to continuously provide real-time information about bandwidth usage. The produced characteristics include information about short-term peaks and about the percentage of bandwidth used by different protocols in different layers of the OSI model hierarchy, including detection of application protocols that use dynamic ports.

Keywords: performance monitoring, end-to-end performance, bandwidth measurement, passive monitoring.

1 Introduction

For network planning and troubleshooting it is useful to know what is the load on network links including its dynamics in different time scales and distribution into protocols in different layers of the OSI hierarchy.

There are several terms related to link load. *Installed bandwidth* is a property of the link technology, such as 1 Gb/s for Gigabit Ethernet. *Used bandwidth* is part of the installed bandwidth occupied by traffic. It is generally changing in time and we usually quantify it by average values over certain time periods. *Available bandwidth* is a complement of used bandwidth to installed bandwidth. As such it is also usually changing in time. The term *capacity* is sometimes used instead of bandwidth. Electrical engineers use the term bandwidth to express the width between the lowest and highest frequency in signal spectrum. In computer science bandwidth is commonly used to denote the number of bits transferred per second.

Achievable throughput is a different metrics, which de-

notes the maximum volume of additional data that can be transferred by a transport protocol over certain network path, which usually already carries other traffic. Since most of traffic in Internet is carried by TCP, which is an elastic transport protocol, achievable throughput is higher than available bandwidth and depends on transport protocols competing for bandwidth.

2 Related work

The following methods can be used to monitor characteristics related to link load:

- Active monitoring - stress test
- Active monitoring - lightweight test
- SNMP and Netflow monitoring
- Passive monitoring

An example of a tool for active monitoring stress test is *iperf* [1]. It measures achievable throughput over a network path between two PCs. As this test affects existing user traffic, it cannot be done frequently and it produces different metrics than available bandwidth.

Lightweight active monitoring tools send only a few carefully scheduled packets and analyze the change of their inter-packet spaces after traversal over a network path in order to estimate either path installed bandwidth or path available bandwidth (of the slowest link or the bottleneck link in a path). These tools produce only estimates and tend to be unreliable in high-speed multi-gigabit networks. Examples of tools in this category are *Pathrate* and *Pathload* [2].

Used bandwidth on a link can be reliably monitored by reading router interface byte counters via SNMP. However, routers update their interface counters in a low priority task with unstable delay of several seconds. Therefore, the shortest interval to compute an average link load using this method is about 20 seconds. When we try to read interface counters more frequently, we get distorted results.

For instance, the upper part of Fig. 1 shows 60-second SNMP samples and the lower part shows 1-second SNMP samples. There is a lot of aliasing in 1-second samples due

to interference between counter updates and counter readings. This figure is for illustration only and the upper and lower part do not show the same time period.

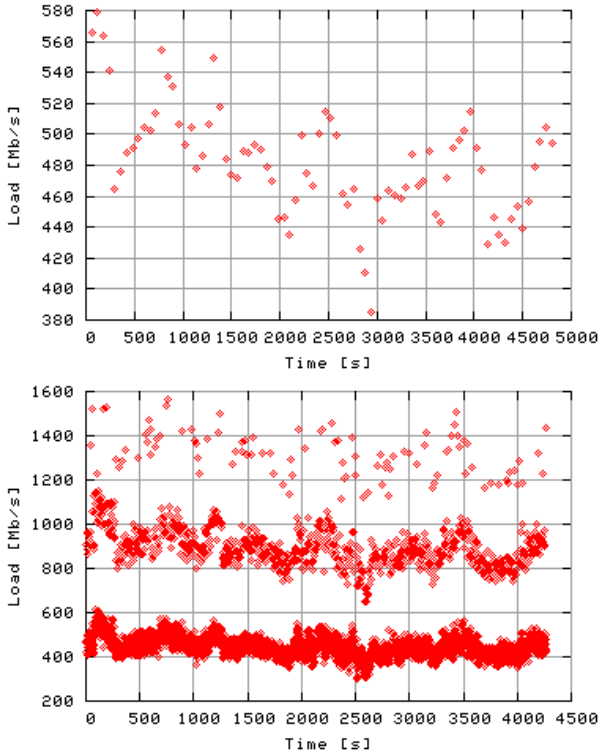


Figure 1. 60-second SNMP samples (above) and 1-second SNMP samples (below)

Netflow [3] records collected from routers can be used to measure the volume of traffic belonging to individual protocols based on their port numbers. However, the Internet traffic is increasingly dominated by protocols that use dynamic ports. For instance when we try classify traffic based on well-known ports of the commonly used application protocols HTTP, HTTPS and FTP, we often end up with most of the traffic unrecognized, shown by the light grey region in Fig. 2.

The rest of this paper is organized as follows. We describe the architecture our passive bandwidth monitoring application in section 3. We give information about the current deployment and availability of the application in section 4. Finally, we provide examples of use in section 5.

3 Architecture

Passive monitoring can be used to monitor used bandwidth and it has several advantages over the other methods:

- Existing user traffic is not affected in any way

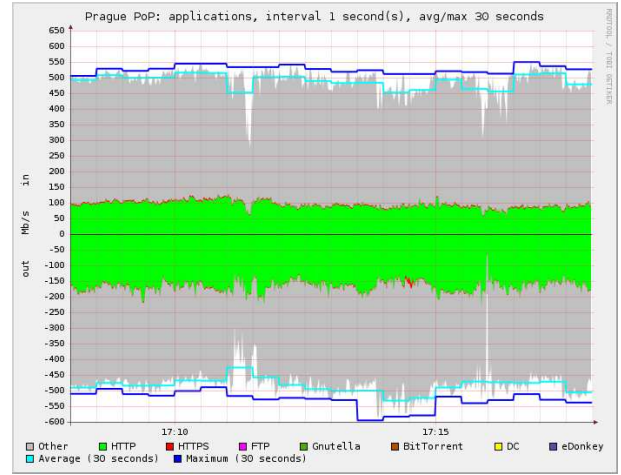


Figure 2. Unrecognized traffic of protocols using dynamic ports

- Monitoring can be continuous
- Monitoring can provide results in frequent intervals
- We can provide used bandwidth by individual protocols and applications

Long-term averages may show the link as lightly loaded, but traffic added to the link can still experience a lot of congestion losses due to short-term peaks of high utilization. In order to get a more accurate view on link utilization, we need information about short-term traffic dynamics.

A combination of header filtering and effective payload searching is needed to classify traffic belonging to applications that use dynamic protocols.

ABW stands for *available bandwidth*, which is what we ultimately want to monitor. Of course, it is only possible to measure used bandwidth directly and we have to compute available bandwidth as a complement to installed bandwidth. ABW is written on top of DiMAPI (Distributed Monitoring Application Interface) [4] and the trackflib library [5].

DiMAPI

DiMAPI is a library to program portable monitoring applications in high level of abstraction. A structure of an application is shown in Fig. 3. An application creates one or more *flows*. Each flow is initially all packets arriving to a specified network interface or a set of network interfaces (in the latter case it is called a *scope*). These network interfaces can be on a local computer or on different remote computers (hence the meaning of distributed in DiMAPI).

```

fd = mapi_create_flow("host1:eth1,host2:/dev/dag0");
fid1 = mapi_apply_function(fd, "BPF_FILTER",
    "net 192.168.0.0/16");
fid2 = mapi_apply_function(fd, "TRACK_FTP");
fid3 = mapi_apply_function(fd, "PKT_COUNTER");

while (1) {
    cnt = (unsigned int*) mapi_read_results(fd, fid3);
    /* ... */
    sleep(1);
}

```

Figure 3. Structure of DiMAPI-based application

The application then applies a sequence of *monitoring functions* to each flow. The order of monitoring functions determines the resulting functionality. Predefined monitoring functions are available for various monitoring primitives. For instance, BPF_FILTER passes only packets matching a specified header filter, TRACK_FTP passes only packets belonging to FTP connections and PKT_COUNTER counts the number of packets. A user can also program new monitoring functions. Finally, we can periodically request results of applied monitoring functions.

DiMAPI can run on top of different network adapters. Currently, regular NICs (Network Interface Cards), DAG cards [6] and COMBO cards [7] are supported. DiMAPI can automatically utilize hardware support in specialized monitoring adapters for certain monitoring functions, transparently to the application. This is allowed by separate implementations of some monitoring function in DiMAPI for different monitoring adapters. After all functions are applied to flows, when the application starts monitoring, DiMAPI checks what functions can be implemented with hardware support based on what network adapters are used and based on the requested order of monitoring functions.

ABW

The ABW application architecture is shown in Fig. 4. Packets are tapped from a monitored link (1) by an optical splitter or by a mirroring port on a router or switch. There are many trade-offs between choosing an optical splitter or a mirroring port [8].

Packets are captured by a network adapter (2), which can be a regular NIC or a specialized monitoring adapter (such as a DAG or COMBO card). Such adapters provide hardware support for some monitoring functions and they can also read packets into computer memory much more efficiently than regular NICs.

Packets are then processed by DiMAPI (3), which is divided into mapid and mapicommd daemons running on remote monitoring stations and the DiMAPI stub, which is linked together with an application.

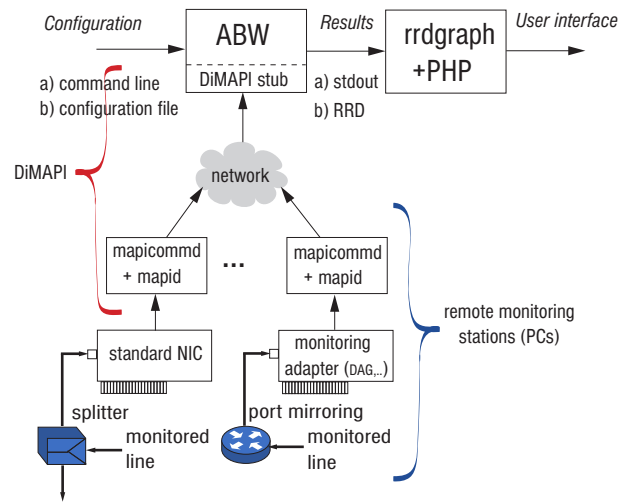


Figure 4. ABW application architecture

The executable of the ABW application (4) reads a configuration file, which specifies what protocols on what remote monitoring stations should be monitored. It also specifies other monitoring parameters, such as frequency of computing the link load and limiting the monitored traffic to a subset of all traffic by header filtering or payload searching. Configuration can also be specified by command-line arguments for debugging. The syntax of the ABW configuration file is similar to the syntax of Windows configuration files (e.g., windows.ini) and its structure is based on concepts of flows and scopes in DiMAPI.

Results are stored in the RRD database or printed on the standard output for debugging. A set of PHP scripts and the rrdgraph utility (5) are used to provide user interface for the application. When monitoring both directions of some monitored link, ABW can accept traffic for each direction from different ports on a multiple-port monitoring adapter or from different NICs on the same monitoring station or even from different monitoring stations. This is all transparent to the user. ABW can also transparently present results from a mixture of MPLS and non-MPLS links, which each require different header filtering to separate individual protocols.

Protocol tracking

Application protocols are detected by the trackflib library, which is a part of DiMAPI and which provides the TRACK monitoring function to request classification into known application protocols. The trackflib library uses a combination of header filtering and payload searching to distribute packets into protocols. Patterns that need to be matched in payload for some protocols are usually close to the beginning of payload and the search can thus finish

Gnutella	DC++
BitTorrent	WEB
eDonkey	FTP
Skype	IP-in-IP

Table 1. Protocols recognized by the trackflib library

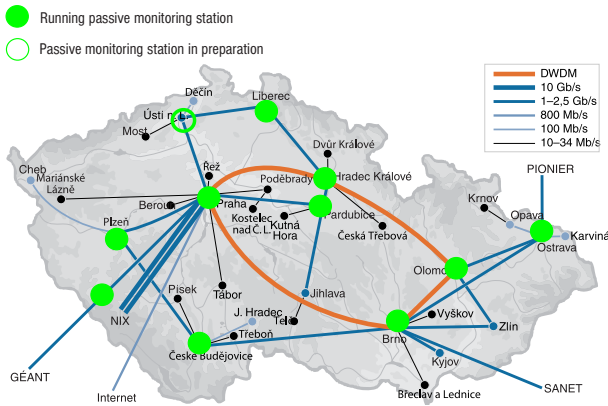


Figure 5. ABW deployment in CESNET

soon. The protocols currently recognized by the trackflib library are listed in Table 3. Note that WEB is different than just matching ports 80 and 443, because these ports are sometimes used by other applications and FTP includes ports used by passive FTP connections.

4 Deployment

We installed 10 passive monitoring stations in the CESNET network, which is NREN (National Research and Educational Network) in Czech Republic. One station monitors the access link to the European Géant2 network and the remaining stations monitor traffic in major backbone nodes of the CESNET network. All stations run the ABW application continuously. The position of the monitoring stations is illustrated in Fig. 5. A similar application called apmon has been deployed by ICS-FORTH in several institutions in Greece.

Some stations are equipped with DAG cards (DAG6.2 or DAG8.2 for 10 Gb/s monitoring and DAG4.3GE for 1 Gb/s monitoring). The remaining stations use Intel Gigabit Ethernet NICs. We plan to gradually upgrade stations that monitor high loaded links to monitoring adapters. All monitoring stations run Linux operation system.

Live measurements of bandwidth usage by ABW in CESNET for selected links are available at the following address: <https://perfmon.cesnet.cz/abw-intro.html>.

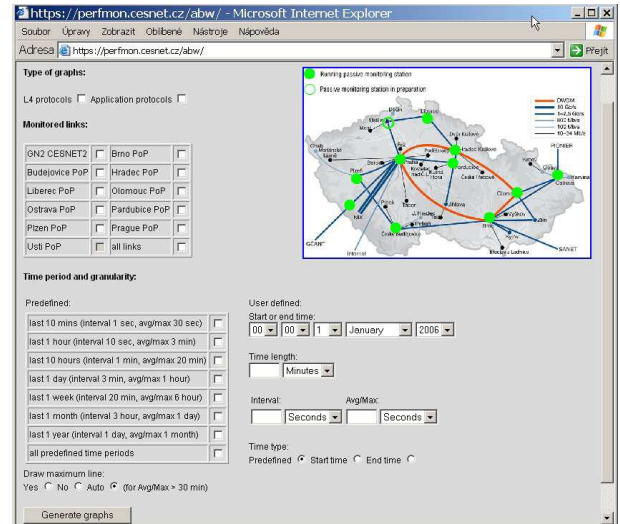


Figure 6. ABW user interface

5 Examples of use

ABW user interface is illustrated in Fig.6. A user can choose two graph types - distribution of L4 protocols (including information about the presence of multicast and IPv6) and distribution of application protocols. More graph types will be added in future.

The user then selects one or more monitoring stations and one or more time periods and granularities for which the network characteristics should be computed from the data in the RRD database and plotted. The time period determines the start and end time plotted in the graphs. The time granularity determines a step of the graph, for which the average or maximum values are computed. The user can either choose from predefined time ranges and granularities or specify any other time period and granularity using a simple form.

Some parameters can also be selected, such as whether the line for maximum values should be plotted or not. Maximum values are sometimes high and reduce readability of average values in the same graph.

Example 1 An example graph produced by ABW that shows distribution of L4 protocols is shown in the upper part of Fig. 7. The characteristics are always computed for two time granularities - fine grain, which is used to plot the main body of the graph and coarse grain, which is used to plot envelope lines in the same graph for comparison. In this particular graph the main body shows fine-grain 1-second averages, whereas the envelope lines shows coarse grain 30-second averages (light blue or light grey line) and maximum (dark blue or dark grey line). You can see that there were frequent short-term peaks of high load present

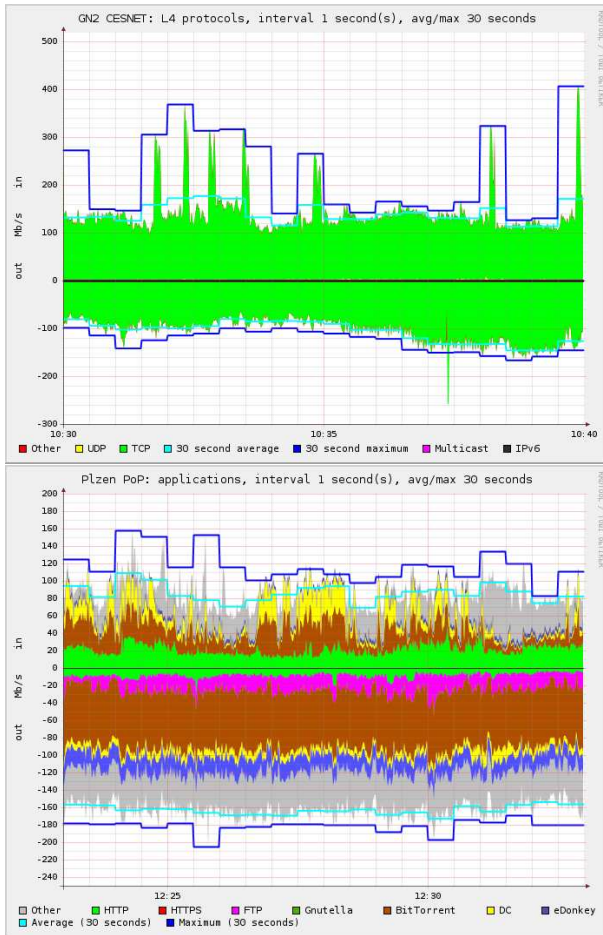


Figure 7. Distribution of L4 protocols (above) and application protocols (below)

on the link, while the coarse-grain averages, which are comparable to what we can get from SNMP monitoring, suggest that the link is seemingly much less loaded. Regarding L4 protocols, the link was dominated by TCP over the plotted period, which is now a common case in the Internet.

Example 2 An example graph produced by ABW that shows distribution of application protocols is shown in the lower part of Fig. 7. You can see that several application protocols that use dynamic ports have been detected. Detection of these protocols provides much better understanding of composition of link usage. This example graph of application protocols does not show the same time range as the example graph of L4 protocols.

We are monitoring CPU load on all monitoring stations. A station with Xeon 3.0 GHz CPU is able to process approximately 400 Mb/s of sustained traffic with Intel Gigabit Ethernet NIC before becoming overloaded.

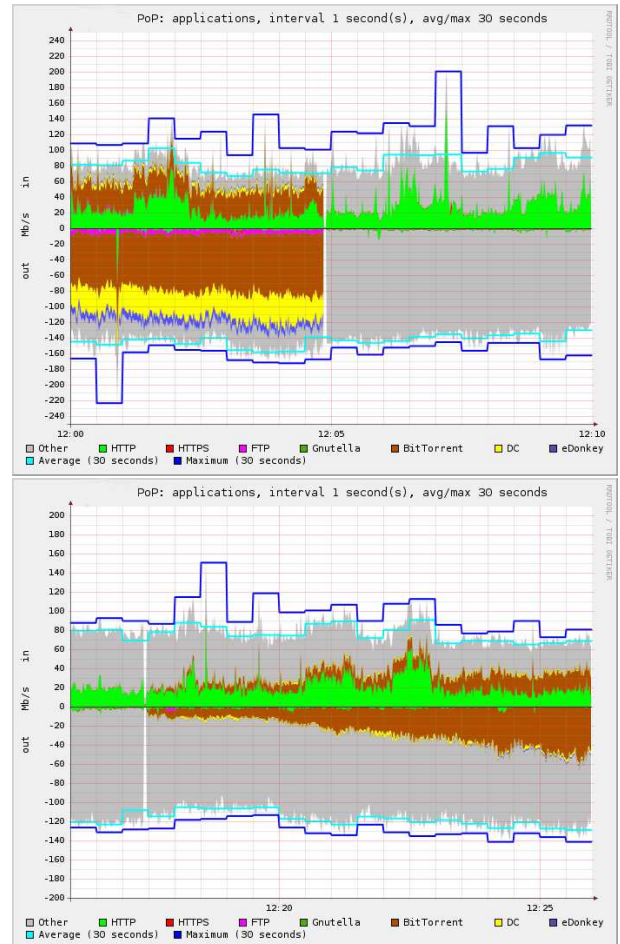


Figure 8. The situation when we stop (above) and start again (below) monitoring of protocols that use dynamic ports

Example 3 When we stop monitoring of protocols that use dynamic ports, we lose much of the information about protocol distribution, see the upper part of Fig. 8. When we start to monitor these protocols again, it takes some time until we get information about protocols distribution, see the lower part of Fig. 8. The reason is that we usually need to capture the beginning of a control connection that negotiates dynamic ports for the data connection. We cannot classify dynamic ports of connections that started in the past.

6 Conclusion

We have developed a passive monitoring application for non-intrusive continuous monitoring of link load, its distribution into protocols and applications and its short-term dynamics.

Contrary to other approaches, we can provide informa-

tion about traffic dynamics in short time scales, showing short peaks of load, which are invisible in coarse-grain monitoring, but affecting throughput of added traffic.

We can also provide detailed information about distribution of link load into protocols in different layers of the OSI hierarchy, including application protocols that use dynamic ports.

The application can automatically benefit from using hardware monitoring adapters, but it can also run without modifications using regular NICs.

In our future work, we want to quantify traffic burstiness at packet level and to plot real-time distribution of burst sizes and its time evolution in three-dimensional graphs. There are several approaches to quantify traffic burstiness (e.g.,[12]).

Acknowledgements

The ABW application has been developed as part of the JRA1 activity [9] of the GN2 project. DiMAPI and the trackflib library were developed by the LOBSTER project (Contract No. 004336) [10]. DiMAPI is based on MAPI, which was developed by the SCAMPI project (Contract No. IST-2001-32404) [11].

References

- [1] Iperf. <http://dast.nlanr.net/Projects/Iperf>.
- [2] Manish Jain, Constantinos Dovrolis. Pathrate and pathload tools, <http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/bw.html>.
- [3] Netflow, <http://www.cisco.com/warp/public/732/netflow>.
- [4] DiMAPI, <http://mapi.uninett.no>.
- [5] Demetres Antoniadis, *Appmon: An Application for Accurate per Application Network Traffic Characterisation*, submitted for Broadband Europe, December 2006, Geneva, Switzerland.
- [6] DAG cards, Endace corporation, <http://www.endace.com>.
- [7] Liberouter project, <http://www.liberouter.org>.
- [8] Sven Ubik, *Optical splitters vs. mirroring ports*, JRA1 GN2 activity working document, http://wiki.perfsonar.net/jra1-wiki/index.php/Passive_Monitoring_Installation.
- [9] JRA1 activity of the GN2 project, <http://www.perfsonar.net>.
- [10] LOBSTER project, <http://www.ist-lobster.org>.
- [11] SCAMPI project, <http://www.ist-scampi.org>.
- [12] Roman Krzanowski, *Burst (of packets) and burstiness*, 66th IETF meeting, July 2006, Montreal, Canada.