

ABW - Short-timescale passive bandwidth monitoring

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

November 18, 2006

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.

Link load is traditionally monitored by reading router interface byte counters via SNMP. This type of monitoring provides only the total load on the link, without distribution into protocols and only as relatively long-term averages.

Routers update their interface counters in a low priority task, resulting in of several seconds, which are fluctuating and unpredictable. Therefore, the shortest interval to compute an average link load is about 30 seconds. When we try to read interface counters more frequently, we get distorted results. For instance, Fig. 1 shows 60-second SNMP samples on the left and 1-second SNMP samples on the right. There is a lot of aliasing in 1-second samples due to interference between counter updates and counter readings. A link may look lightly loaded, but traffic added to the link still experiences 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.

Netflow [1] 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 end up with most of the traffic unrecognized, shown by the grey region in Fig. 2. A combination of header filtering and effective payload searching is needed to classify traffic belonging to such protocols.

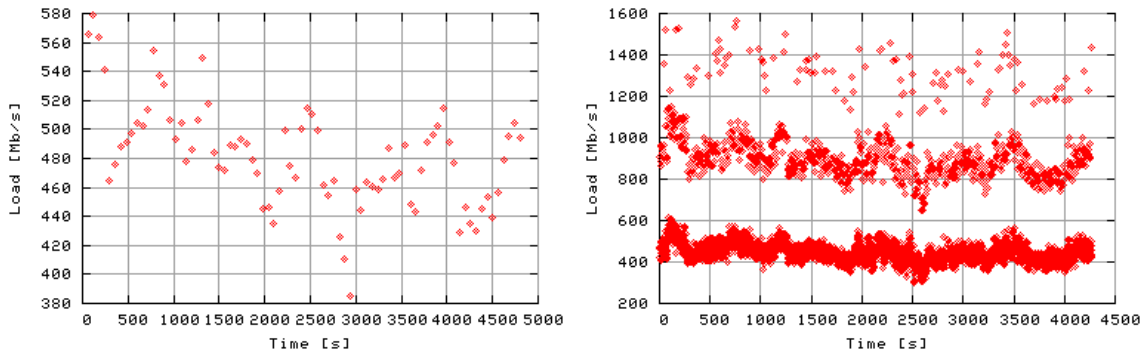


Figure 1: 60-second SNMP samples (left) and 1-second SNMP samples (right)

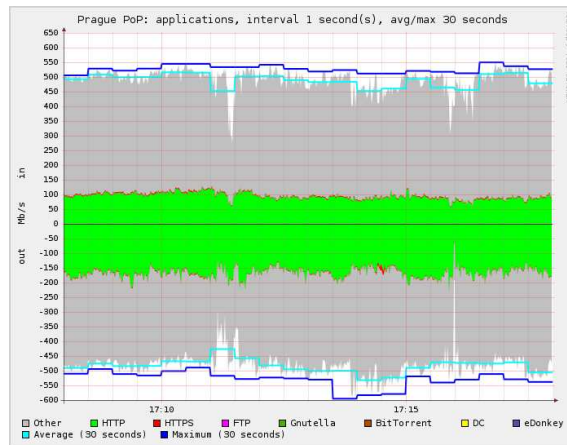


Figure 2: Unrecognized traffic of protocols using dynamic ports

The rest of this paper is organized as follows. We describe the architecture our passive bandwidth monitoring application in section 2. We give information about the current deployment and availability of the application in section 3. Finally, we provide examples of use in section 4.

2 Architecture

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 available bandwidth is a complement to installed bandwidth. ABW is written on top of DiMAPI (Distributed Monitoring Application Interface) [2] and the tracklib library [3].

DiMAPI

DiMAPI is a library to program portable monitoring applications in high level of abstraction. An application opens 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).

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 header filtering (BPF_FILTER), payload searching (STR_SEARCH), counting

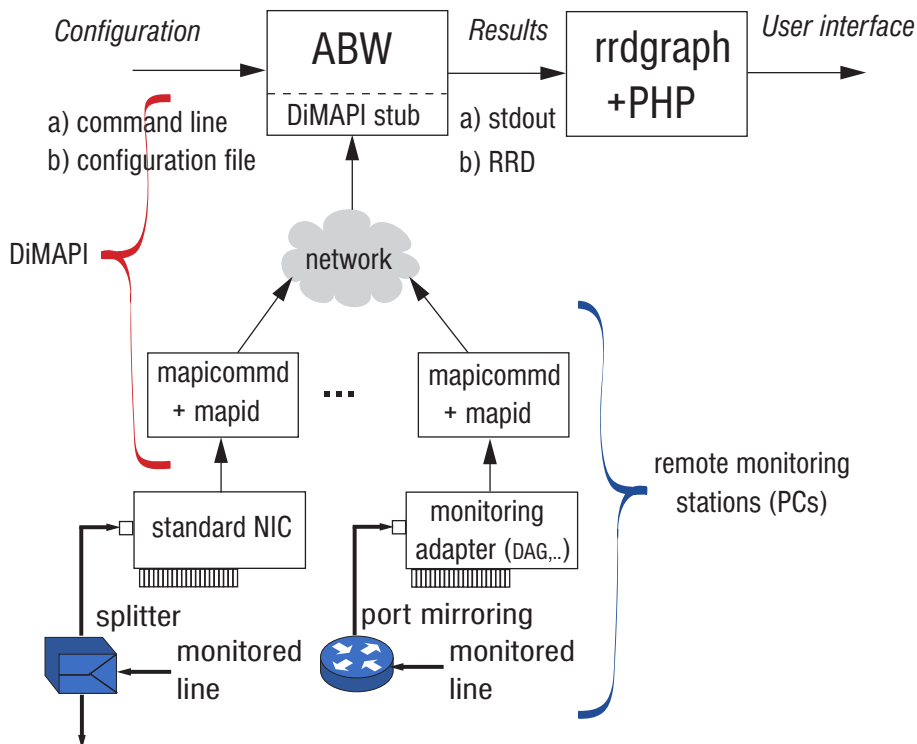


Figure 3: ABW application architecture

statistics (PKT_COUNTER) and for other monitoring primitives. A user can also program new monitoring functions.

DiMAPI can run on top of different network adapters. Currently, regular NICs (Network Interface Cards), DAG cards [4] and COMBO cards [5] are supported. DiMAPI can automatically utilize hardware support in specialized monitoring adapters for certain monitoring functions, transparently to the application. This is enabled 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. 3. 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 [6].

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.

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

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

Table 1: Protocols recognized by the trackflib library

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.q., 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 soon. The protocols currently recognized by the trackflib library are listed in Table 2. 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.

3 Deployment and availability

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. 4. A similar application called appmon has been deployed 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.

Availability

ABW is currently distributed under the BSD license as part of DiMAPI. The DiMAPI code can be downloaded from an SVN server [2].

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

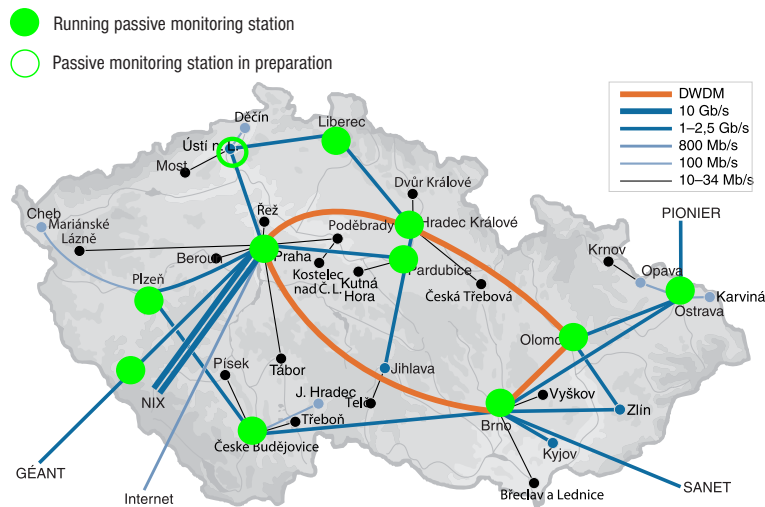


Figure 4: ABW deployment in CESNET

Monitoring of the GN2 - CESNET link is accessible to the public. Monitoring of CESNET backbone links requires authorization by username and password.

4 Examples of use

ABW user interface is illustrated in Fig.5. 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 ranges and granularities for which the network characteristics should be computed from the data in the RRD database and plotted. The time range 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.

An example graph produced by ABW that shows distribution of L4 protocols is shown in the left part of Fig. 6. 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 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.

An example graph produced by ABW that shows distribution of application protocols is shown in the right part of Fig. 6. 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 also monitoring CPU load on all monitoring stations. A station with Xeon 3.0 GHz

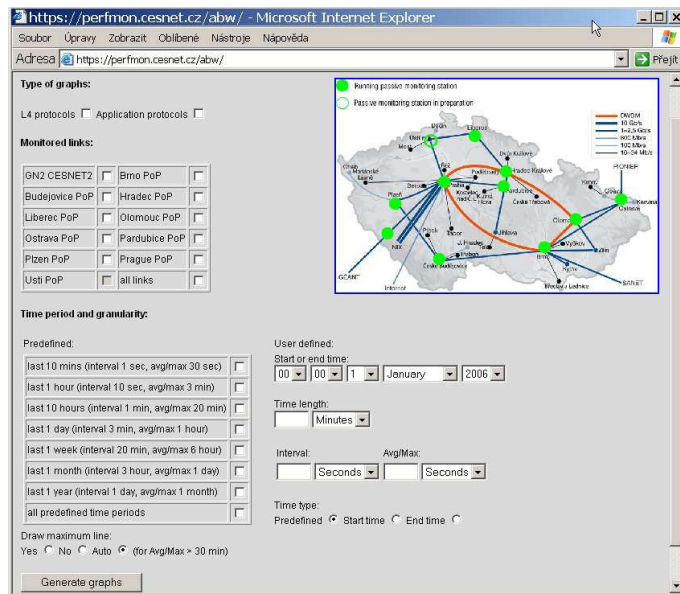


Figure 5: ABW user interface

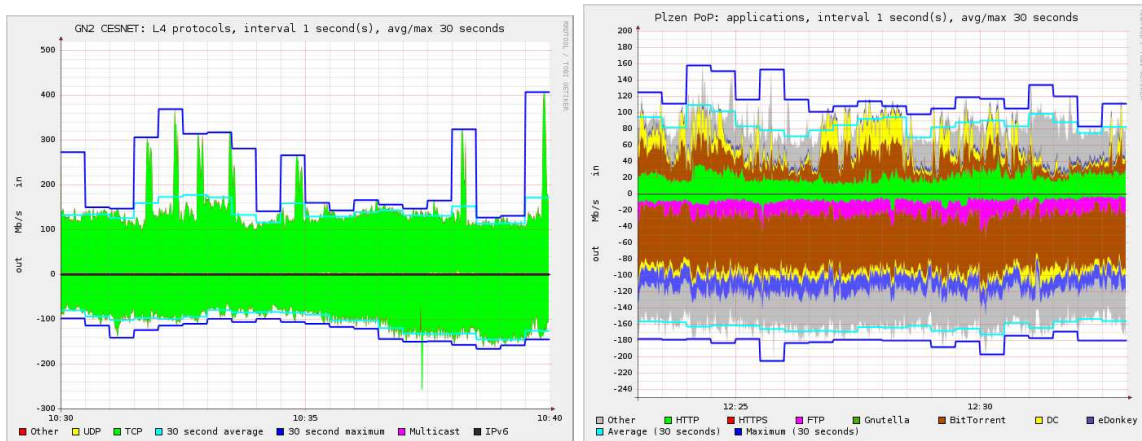


Figure 6: Distribution of L4 protocols (left) and application protocols (right)

CPU was able to process approximately 300 Mb/s of sustained traffic with Intel Gigabit Ethernet NIC and approximately 700 Mb/s of sustained traffic with DAG 4.3GE card. In this case, the DAG card provides advantage only by its more effective data transfer. Hardware filtering in the card was not used.

5 Conclusion

We have developed a passive non-intrusive application for continuous monitoring of traffic composition and dynamics on network links. Contrary to previous approaches, we can provide detailed information about distribution of link load among protocols in different layers of the OSI hierarchy, including application protocols that use dynamic ports. We can also provide fine-grain information about traffic dynamics in short time scales, which can reveal short and high peaks of load, which are invisible in coarse-grain monitoring. The application can automatically benefit from using hardware monitoring adapters, but it can also run without modifications using regular NICs.

In our future research, we want to measure traffic burstiness at packet level exactly 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.,[10]).

Acknowledgements

The ABW application has been developed as part of the JRA1 activity [7] of the GN2 project. DiMAPI and the trackfib library were developed by the LOBSTER project (Contract No. 004336) [8]. DiMAPI is based on MAPI, which was developed by the SCAMPI project (Contract No. IST-2001-32404) [9].

References

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