

ABW – Short-timescale passive bandwidth monitoring

**Sven Ubik (CESNET), Demetres
Antoniades (ICS-FORTH), Arne Oslebo (UNINETT)**

7.12.2006

1 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.

2 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, Figure 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.

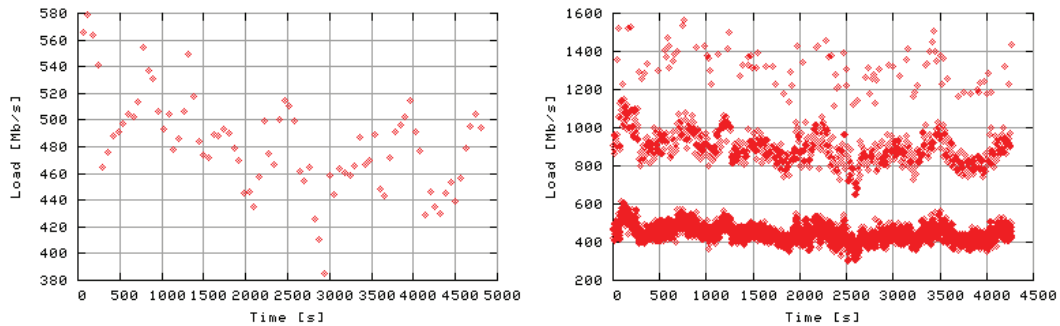


Figure 1: 60-second SNMP samples (left) and 1-second SNMP samples (right)

Netflow¹ 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 to 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 Figure 2. A combination of header filtering and effective payload searching is needed to classify traffic belonging to such protocols.

The main advantages of the ABW application when compared to SNMP and Netflow monitoring are the following:

- We can distinguish bandwidth used by different protocols in different layers of the OSI hierarchy (L2, L3, L4 or application protocols)
- We can monitor bandwidth usage in short intervals (e.g., 1 second) and thus detect short peaks

Detecting application protocols requires a combination of header filtering and payload searching.

The rest of this report is organized as follows. We describe the architecture of our passive bandwidth monitoring application in Section 3. We give information

¹<http://www.cisco.com/warp/public/732/netflow>

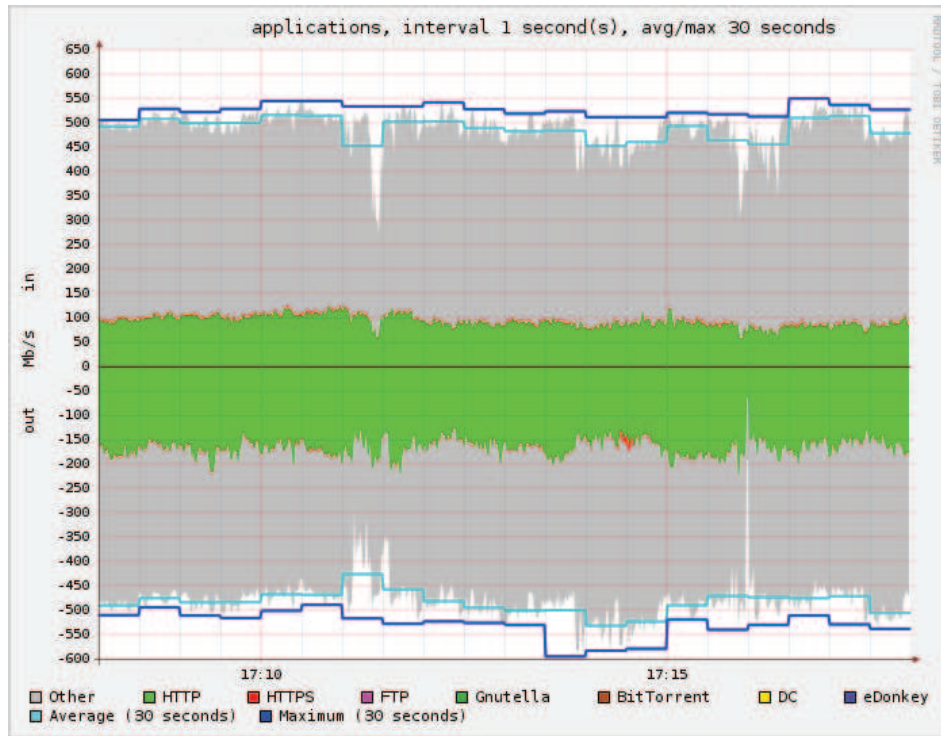


Figure 2: Unrecognized traffic of protocols using dynamic ports

about the current deployment and availability of the application in Section 4. Finally, we provide examples of use in Section 5.

3 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 DiMAPI² (Distributed Monitoring Application Interface) and the trackflib library [Ant06] as a C-language executable and a set of PHP and rrdtool scripts.

3.1 ABW

The ABW application architecture is shown in Figure 3. Packets are tapped from a monitored link (1) by an optical splitter or by a mirroring port on a router or switch.

²<http://mapi.uninett.no>

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 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.

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

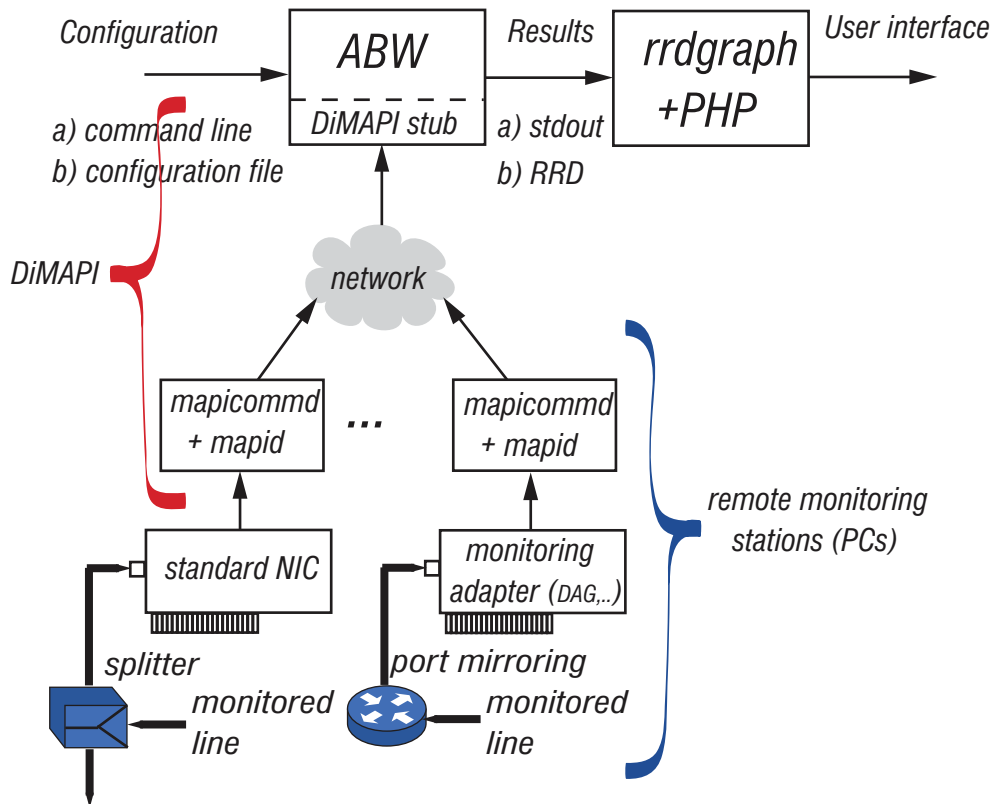


Figure 3: ABW application architecture

DiMAPI can run on top of different network adapters. Currently, regular NICs (Network Interface Cards), DAG cards³ and COMBO cards⁴ 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.

DiMAPI (Distributed MAPI) is an extension to MAPI. The difference between DiMAPI and MAPI is in communication between the stub library linked with an application and the daemons running on a monitoring station. In DiMAPI this communication takes place over a TCP connection and `mapid` and `mapicommd` daemons can run on different machines than the application. The `mapicommd` daemons mediate network communication with the `mapid` daemon. On the other hand, MAPI uses a UNIX socket, `mapicommd` daemon is not used and `mapid` daemon must run on the same machine as the application.

3.3 Local and distributed mode

ABW can be used in a local or distributed mode. In the local mode, ABW uses MAPI. Each remote monitoring station runs a local copy of the ABW executable and the `mapid` daemon. The web server contacts remote monitoring stations directly. This configuration is illustrated in Figure 4. Arrows show directions of data flow.

In distributed mode, ABW uses DiMAPI. A central station runs the ABW executable, which gathers results from remote monitoring stations running the `mapid` and `mapicommd` daemons. The web server contacts only the central station. This configuration is illustrated in Figure 5.

ABW can also use a combination of these two modes. That is the web server can communicate with some remote monitoring stations directly and obtain results from other monitoring stations through the central station. You simply point the PHP scripts on the web server to the right station for each monitored link to the central station or to a remote monitoring station.

Using the central station makes configuration on the web server simpler, but contacting the remote monitoring stations directly usually provides faster response. Configuration on the web server is simpler and the communication with remote monitoring stations is more effective when we can group monitoring of multiple links to one scope. This is only possible when the same sequence of monitoring

³<http://www.endace.com>

⁴<http://www.liberouter.org>

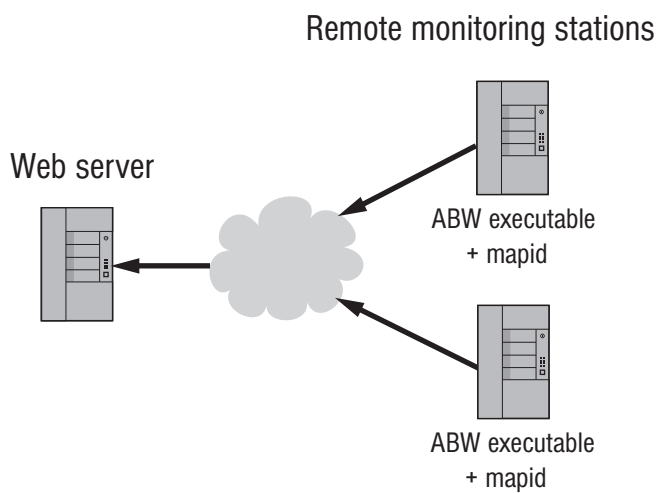


Figure 4: ABW running in the local mode

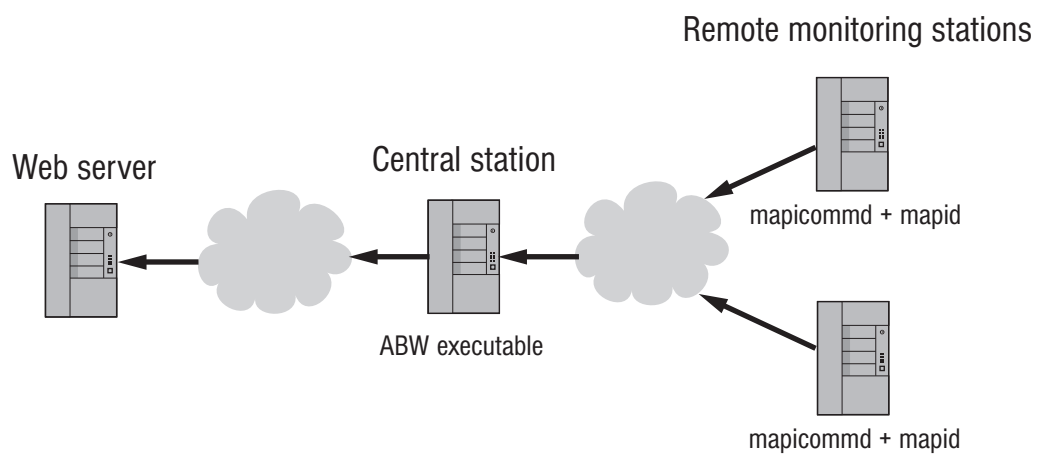


Figure 5: ABW running in the distributed mode

functions can be applied to multiple monitored links. This is not possible when some links have MPLS packets and some links have regular IP packets (because these two cases require different header filtering to identify protocols) or when multiport monitoring adapters are used (because an extra monitoring function is required to separate packets on individual ports).

3.4 Monitoring cards

Each remote monitoring station needs one or more monitoring cards (regular NICs, DAG cards or COMBO cards), which receive traffic tapped from the monitored links (using optical splitters or monitoring ports on routers).

Two directions of one monitored link can be monitored by:

- two ports on a multi-port monitoring card,
- two single-port monitoring cards in one remote monitoring station,
- two single-port monitoring cards in two remote monitoring stations.

This is configured in the ABW configuration file `abw.cfg`. ABW gathers results correctly and displays them in one graph (inbound traffic in the upper part of the graph and outbound traffic in the lower part of the graph).

DAG cards can capture packets at 100% line rate without losses and provide much more effective packet transfer to the memory of the host PC than a regular NIC leaving more CPU capacity to packet processing. When we want to monitor only a subset of traffic on the monitored line, we can define this subset by header filtering in the ABW configuration file. When a DAG card with IPF coprocessor is used (such as DAG4.3GE), this header filtering is implemented in the hardware of the monitoring card and only requested packets are forwarded to the host PC. Support of newer DAG cards with DSM classification will be available in the future version.

3.5 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 1. Note that WEB is different than just matching ports 80 and 443, because these

Ethernet NICs. We plan to gradually upgrade stations that monitor high loaded links to monitoring adapters. The first such station will be upgraded to two DAG8.2 cards early 2007. All monitoring stations run Linux operation system.

Live measurements of bandwidth usage by ABW in CESNET are available at the following address: <https://perfmon.cesnet.cz/abw-intro.html>. Monitoring of the GN2 - CESNET link is accessible to the public. Monitoring of CESNET backbone links requires authorization by username and password.

5 Examples of use

ABW user interface is illustrated in Figure 7. 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. The user then selects one or more monitored links 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 periods 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 left part of Figure 8. 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.

Example 2

An example graph produced by ABW that shows distribution of application protocols is shown in the right part of Figure 8. You can see that several

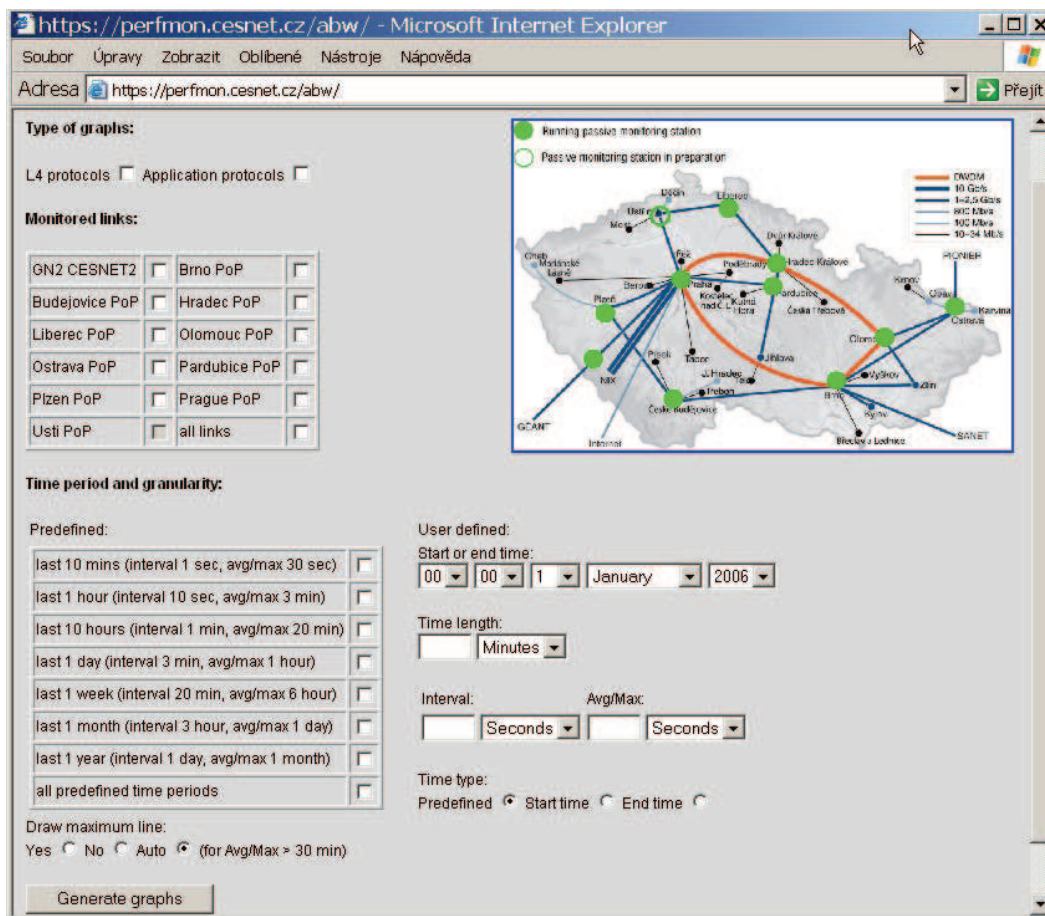


Figure 7: ABW user interface

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.

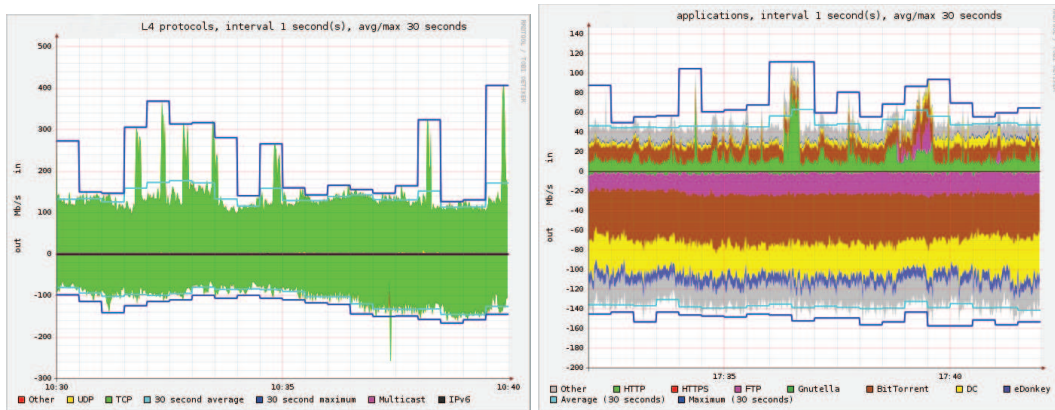


Figure 8: Distribution of L4 protocols (left) and application protocols (right)

Example 3

When we stop monitoring of protocols that use dynamic ports, we loose much of the information about protocol distribution, see left part of Figure 9. When we start to monitor these protocols again, it takes some time until we get information about protocols distribution, see the right part of Figure 9. 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 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.

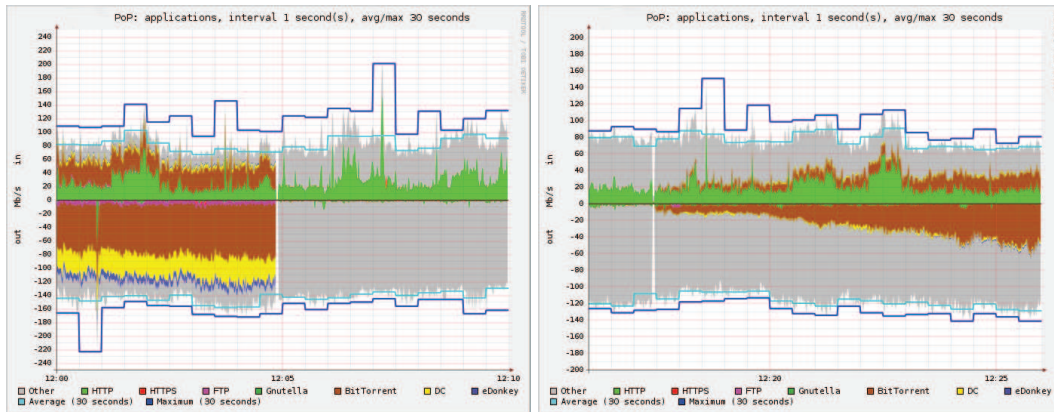


Figure 9: The situation when we stop (left) and start again (right) monitoring of protocols that use dynamic ports

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., [Krz06]).

6.1 Acknowledgements

The ABW application has been developed as part of the JRA1 activity⁵ of the GN2 project. DiMAPI and the trackflib library were developed by the LOBSTER project⁶ (Contract No. 004336). DiMAPI is based on MAPI, which was developed by the SCAMPI project⁷ (Contract No. IST-2001-32404).

References

- [Ant06] Antoniadis D.: Appmon: An Application for Accurate per Application Network Traffic Characterisation, submitted to *Broadband Europe*, December 2006, Geneva, Switzerland.
- [Ubi07] Ubik S.: *Optical splitters vs. mirroring ports*, JRA1 GN2 activity working document, 2007. Available online⁸.
- [Krz06] Krzanowski, R.: Burst (of packets) and burstiness, *66th IETF meeting*, July 2006, Montreal, Canada.

⁵<http://www.perfsonar.net>

⁶<http://www.ist-lobster.org>

⁷<http://www.ist-scampi.org>

⁸http://wiki.perfsonar.net/jra1-wiki/index.php/Passive_Monitoring_Installation