Network monitoring in the Tier2 site in Prague

Marek Eliáš, Lukáš Fiala, Jiří Horký, Jiří Chudoba, Tomáš Kouba, Jan Kundrát, Jan Švec

E-mail: elias@fzu.cz
Institute of Physics, AS CR, Na Slovance 22, 182 21, Prague, Czech Republic

Abstract. Network monitoring provides different types of view on the network traffic. It's output enables computing centre staff to make qualified decisions about changes in the organization of computing centre network and to spot possible problems. In this paper we present network monitoring framework used at Tier-2 in Prague in Institute of Physics (FZU). The framework consists of standard software and custom tools. We discuss our system for hardware failures detection using syslog logging and Nagios active checks, bandwidth monitoring of physical links and analysis of NetFlow exports from Cisco routers. We present tool for automatic detection of network layout based on SNMP. This tool also records topology changes into SVN repository. Adapted weathermap4rrd is used to visualize recorded data to get fast overview showing current bandwidth usage of links in network.

1. Motivation

Network monitoring is essential for keeping a network functional, fast and manageable. It is needed to track prefailure and failure notices of networking hardware as it is critical in many deployments. It's outputs are used when making decisions about a reorganization of topology and network upgrades. When it includes connection tracking it can be very useful in security analysis. In addition it can be used for comparing data transfer requirements of different experiments for traffic from storage elements as well as from collaborating sites.

2. Different views of network monitoring

Network monitoring can be viewed from different sides. In Institute of Physics in Prague (FZU), we consider three main aspects of network monitoring. Firstly, we focus on the network equipment itself. We monitor health of the equipment and track changes of it's configuration. On the higher level we focus on physical links, mostly on their bandwidth usage, up/down state and the topology of the whole network. Last but not least we collect and analyse netflow data. This enables us to track individual network connections, monitor amount of traffic transfered between different parts of network and data exchange between local network and outside world.

3. Low level monitoring of network equipment

To keep track of a functionality of the network equipment itself we collect and process log messages from these devices on central syslog and use active checks of their health.

3.1. Event logging to central syslog
Routers, ethernet and fibre channel switches are configured to send log messages to central syslog where they are stored and processed if needed. By processing is meant that syslog server propagates every event of priority critical and above through NSCA[1] to passive check sensor in Nagios [1]. On central syslog we run syslog-ng and our setup is similar to approach presented in [2].

This setup is sufficient for us at the moment, but has some issues. Firstly, the event filtering by priority is not very reliable. Not all switches give reasonable priority to the events. For some switches event of regular login of an administrator has level critical — same as the power supply failure event. To prevent occasional false alarms, better filtering rules would be needed, ideally based on knowledge of potential log messages for each switch. Next problems is that switches do not send log messages in some situations like reboot. Usually only messages about bringing up a few last ports arrive to syslog when a switch is booting, so it is hard to deduce that a switch was rebooted from log messages only. We have solved this problem by active checks mentioned later.

3.2. Active checks in Nagios
We use combination of active check sensors to monitor running state and short reboots of switches. First is a ping sensor which pings on management interface of the switch. Second is uptime sensor which gets switch uptime value through SNMP and checks whether it is larger than checking period (otherwise reboot occurred during last period). In case of rack switches, this sensor turned out to be a useful point where one can spot problems of a whole rack like short power interruptions which probably rebooted whole rack of worker-nodes not only the switch. There is a problem in uptime monitoring when uptime overflows. For most switches this occurs very seldom, but we have some switches which drop down uptime every four months without being rebooted.

4. Monitoring of physical network links and network topology
Current combination of MRTG [3], weathermap4rrd [4] and our tool fzu_weathermap enables us to detect network topology automatically, view topology of the network with current saturation of links and view both in any point in the history through Historical view feature. However, historical view reads link saturation data from RRD files of MRTG which agregates older data, so short times anomalies in traffic get lost after some time.

4.1. Configuration of MRTG and weathermap4rrd
Only configuration done to already deployed MRTG was converting stored data to RRD files to collaborate with weathermap4rrd. A PHP version of weathermap4rrd generates pictures of network on demand and is very slow when drawing large networks. A Perl version is much faster and generates images on regular basis (run from cron). The Perl version missed many features of the PHP version like showing a custom image when a mouse is over, or providing access to further information about a link, so we added missing features to the perl version and deployed this modified version.

4.2. fzu_weathermap
Our tool fzu_weathermap forms a glue between MRTG and weathermap4rrd and takes care of a network topology. It is run periodically from cron and generates configuration files for weathermap4rrd. It configures weathermap4rrd to take values from right RRD files, show daily MRTG graph when a mouse is over a link and to show all MRTG graphs when clicking on it.

[1] Nagios Service Check Acceptor — used by Nagios to receive results of passive checks

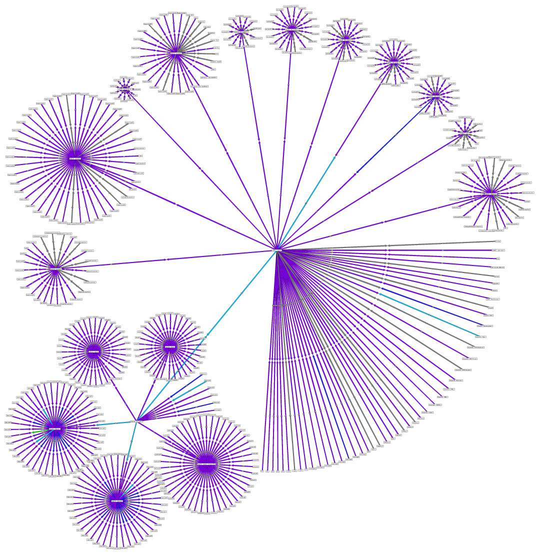


Figure 1. Network view of fzu_weathermap

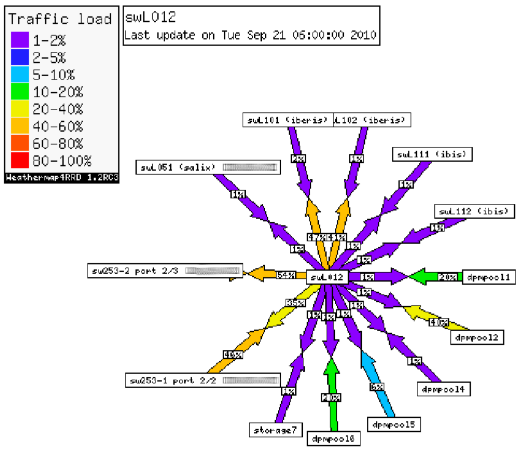


Figure 2. Switch view of fzu_weathermap

Currently there are two kinds of generated maps: map of a whole network (figure 4) and separate maps for each switch showing only one switch and devices connected directly to its ports (figure 2).

Fzu_weathermap works by collecting topology data from SNMP. It gets SNMP description of each port on every switch, generates graph of network topology and saves it in graphviz [5], [6] dot format. First word of the port description should contain unique name of a node connected to the port potentially suffixed by number of this link in LACP bonding (ie. suffix '-2' if it is link number 2 in bonded link). Backup and other kinds of duplicate links are supported by similar suffixes. Network topology is updated every 15 minutes and changed dot files are committed into subversion repository. These files are then processed by graphviz layouter circo which assigns coordinates to nodes on the map. These coordinates are used to generate weathermap4rrd configuration files in where each node must have preset coordinates.

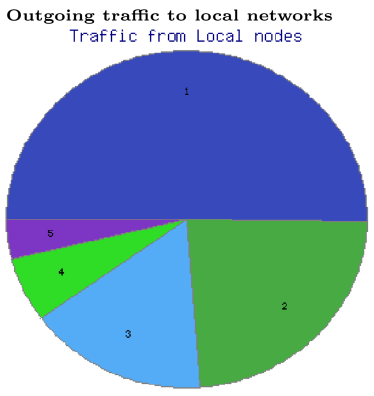
In addition to drawing maps of current state of a network, fzu_weathermap has a feature of drawing state in any point in the past. It works by checking out state of network topology in requested time from SVN repository and executing weathermap4rrd with the time option. This option forces weathermap4rrd to get link saturations in specific time from RRD files instead of last values. This makes fzu_weathermap applicable in post mortem as well as real time analysis of network problems.

4.3. Some considerations about deployment of fzu_weathermap

As stated above, fzu_weathermap gets network topology from port descriptions through SNMP. This means that these port descriptions should be suitably setup. Format for descriptions is flexible enough so one can incorporate in them any other needed information. Only the first word is required to be a unique identifier of the connected node for fzu_weathermap.

Complexity of deployment depends mainly on current state of network documentation. If documentation contains identifiers (ie. DNS name) of devices connected to every port on a switch, there should be no problem to generate configuration commands for the switch to set up port description using common tools.

If such documentation does not exist, one needs to find out first how are the network



no	IP range	octets	%	Nodes
1.	172.16.2/24	144.77 TB	47.92 %	subcluster iberis
2.	172.16.1/24	62.23 TB	20.60 %	subcluster salix
3.	172.16.6/24	58.41 TB	19.34 %	subcluster ibis
4.	172.16.5/24	28.18 TB	9.33 %	subcluster saltix
5.	147.231.25.101 - 147.231.25.172	8.48 TB	2.81 %	subcluster golias
6.	147.231.25/24	15.17 GB	0.01 %	infrastructure nodes
7.	172.16.4/24	13.63 MB	0.00 %	subcluster hypericum
		302.09 TB		All local nodes

Figure 3. Example output of fzu_site_stat

elements connected and fill this information in port descriptions. Fzu_weathermap can keep this information, show current network setup, archive topology changes, show the anomalies and oddities in a setup of the network and help to eliminate them. But unfortunately it is not able to find out where is this or that strange cable connected. For example a MAC address cache of the switch in combination with DHCP server configuration can be very helpful. But when all technological solutions fail, there is no other way than to check the cables manually.

5. Monitoring based on netflow

We collect netflow data from our central Cisco 6500 router on a small dedicated server serving as netflow data collector. We do not observer any significant loss of routing performance in our environment caused by netflow. On netflow collector server we use flow-tools [7] as a netflow data capturing software. We collect all exported data, it takes 3–7GB per month. Flow-tools package contains a data collector as well as command line utilities for manipulation, filtering and summarizing captured flow data.

5.1. FlowViewer

FlowViewer [8] is a web interface based on flow-tools. It is capable of filtering flows and displaying statistics for these flows. In addition, it can plot approximate changes of bitrates of filtered connections in time. Very useful feature is MRTG style regular bandwidth plotting based on rrdtools — one defines a filter and daemon regularly counts approximate bandwidth consumed by connections from filtered flows. Values are then saved to RRD file and the tool periodically generates graphs of bandwidth usage.

5.2. fzu_site_stat monitor

We created this tool to see how many data is routed between nodes, networks, node groups as well as external institutions whose names are automatically discovered from IP addresses through whois service. It regularly creates statistics for the last day or the last month. Network/node groups can be defined manually in a configuration file or (in case of transfers to the world) can be resolved from whois. Statistics have a form of a table and a pie chart (figure 3).

As it is impossible to make whois request for every flow, fzu_site_stat maintains an internal persistent cache of resolved prefixes and subnets. As flows with negligible traffic are ignored (threshold is configurable) and collaborating networks do not change often, about 5–20 whois requests are done in our environment per day with reasonably built whois cache (meant not in the first run). Flow-tools package is used for manipulation with flow data, especially flow-nfilter. This program uses files containing filters specifications and select from the input only

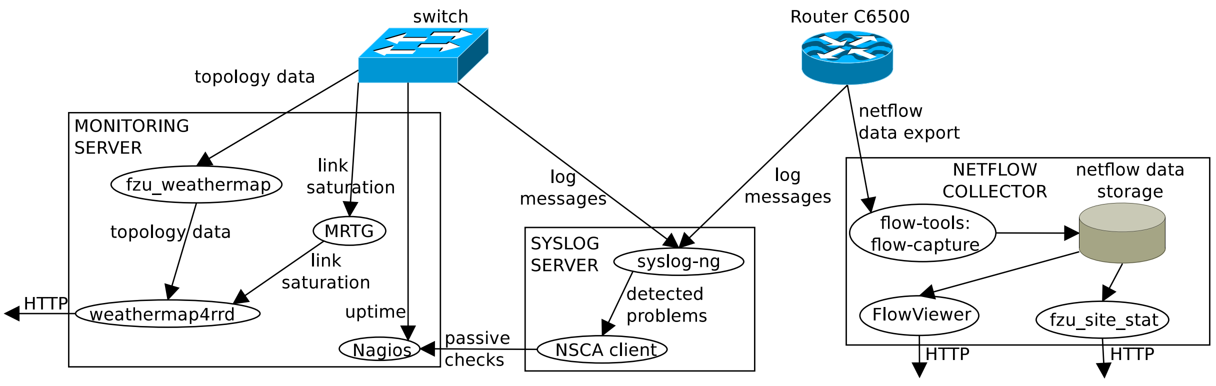


Figure 4. Scheme of network monitoring framework in FZU

flows satisfying filter definition.

Fzu_site_stat is fully configurable. It has a main config file which can possibly contain some predefined overrides for subnetwork names. Each set of statistics is configured by one flow-nfilter filter file which contains several filters and metadata for processing by fzu_site_stat. Subnet overrides can be defined separately for every filter. It is possible to define a group of nodes or a group of ranges to be taken as one account in statistics.

In FZU it is used to create statistics of transfered data between DPM pool nodes and subclusters of worker nodes and between DPM pool nodes and collaborating institutions in the Atlas experiment. Similar statistics are done for SAM stations in the D0 experiment as well as for the whole farm to see which services are most accessed form outside and which institutions are origins or targets of these transfers.

Acknowledgments

This work was supported by Academy of Sciences of the Czech Republic.

References

[1] Nagios Enterprises Nagios `http://www.nagios.org/`
[2] Scheidler B Automated log monitoring with nagios and syslog-ng `http://nordicmeetonnagios.op5.org/op5media/nmn2009/presentations/Balasz%20Scheidler%20-%20syslog-ng-3.0.pdf`
[3] Oetiker T The multi router traffic grapher `http://oss.oetiker.ch/mrtg/`
[4] Fontelle A Weathermap4rrd `http://weathermap4rrd.tropicalex.net/`
[5] Gansner E R and North S C 2000 *SOFTWARE - PRACTICE AND EXPERIENCE* **30** 1203–1233
[6] Ellson J, Gansner R E and North C S Graphviz — graph visualization software `http://www.graphviz.org/`
[7] Fullmer M flow-tools `http://www.splintered.net/sw/flow-tools/`
[8] Loiacono J FlowViewer `http://ensight.eos.nasa.gov/FlowViewer/`