

CESNET Technical Report 15/2009

G3 System – extensions in 2009

TOM KOŠŇAR

Received 11.12. 2009

Abstract

G3 system aims to be a set of complex tools designed for large scale and continuous network infrastructure measurement visualization and reporting. We focused on two areas of system development in 2009 – measurement capabilities of the G3 system especially in the area of virtual infrastructures monitoring and processing efficiency of G3 stand-alone automated visualization tool – the G3 system reporter.

Keywords: network monitoring, infrastructure monitoring, SNMP monitoring

1 Introduction

We use G3 system for complex view on infrastructure of the CESNET2 network – NREN in the Czech Republic. CESNET2 is hybrid network offering services at several layers. It is able to deliver (beside others) dedicated links and infrastructures with specific transport parameters at different levels of virtualization. Many of them are delivered (from the users perspective) as infrastructures created at the CESNET2 IP/MPLS layer and consist either of EoMPLS (Ethernet over MPLS) links or are created as VPLS (Virtual Private LAN Service). Therefore we focused on measurement of virtual channels as well as Class based QoS (Quality of Services).

On the visualization side we observed massive growth of reports periodically generated by G3 system reporter. Many of them are in principle variants of each other visualizing the same sets of objects in different manners (utilization and error state reports of the same network clouds for example). We analyzed the situation and found several way to improve processing efficiency for that cases.

2 Virtual channels measurement

CESNET2 IP/MPLS backbone core is currently based on Cisco technology (CRS-1 and 76xx platforms). The natural starting point could be to adopt current Ciscos' point of view on virtual channels. But we want to keep G3 system vendor and platform independent, so we decided as in other cases to hide everything vendor dependent behind an universal abstraction layer. It is especially the case of identification. The rest of the problem is usually understood as an analogy with classical interface monitoring (with some extensions of course). Even though there is some similarity between view on interfaces and virtual channels we decided to define new object class in G3 system – VCHANNEL.

2.1 MIBs used to measure virtual channels in CESNET2 IP/MPLS backbone

There are currently three MIBs that can be used to measure various types of virtual channel end points in IP/MPLS clouds based on Cisco technology:

- CISCO-IETF-PW-MIB: 1.3.6.1.4.1.9.10.106 – adaptation of RFC5601: Pseudowire (PW) Management Information Base (MIB)
- CISCO-IETF-PW-MPLS-MIB: 1.3.6.1.4.1.9.10.107 – adaptation of RFC5602: Pseudowire (PW) over MPLS PSN Management Information Base (MIB)
- CISCO-IETF-PW-ENET-MIB: 1.3.6.1.4.1.9.10.108 – adaptation of RFC5603: Ethernet Pseudowire (PW) Management Information Base (MIB)

2.2 G3 object identifiers and VCHANNEL identifiers

The most important task is to find out the proper object identification. We insist on being independent on “technological” identifiers like SNMP indexes in G3 measurement system. We define our own identifiers – virtual items that are constructed from a set of pre-processed measured item values. The key point is to select such set of items (within each class), that are resistant to re-indexing or marginal reconstructions on one side and are always exclusive (as a set) on the other side. The process of building appropriate identifier value is always the same in the G3 system - here are the relevant steps with INTERFACE object class example:

Items (OIDs) that have relation to appropriate instance of object (particular INTERFACE for example) may consist of values measured in different parts of global MIB tree – like `ifEntry` and `ifXEntry` in IF-MIB, `ipAddrEntry` in IP-MIB `ordot3StatsEntry` in EtherLike-MIB and so on in case of INTERFACE class of measured object. We call these particular components (sets of OID having the same parent node) sections in G3 system.

Each section must be configured in G3 measurement module to have enough information for successful post-processing. It means that set of OIDs measured (items, values and instance IDs) within each section must enable to re-map and merge section content with others that belong to the same object instance (particular INTERFACE for example). This is done with the help of “technological” identifiers – various SNMP indexes and instance IDs.

Created object instance record contains everything that was measured in current measurement step for particular instance of object class. To get information needed to create absolute identifier for each object we need to import descriptive item values from the whole parent hierarchy into the object record before constructing the identifier. That means (in case of INTERFACE object class example) that descriptive items of measured device SYSTEM object will be imported, because INTERFACE class is defined as child of SYSTEM class within G3 data model. This is common principle of G3 identifier creation mechanism – each object has identifier which is exclusive absolutely. Now we should be able to construct vendor independent, to SNMP re-indexing and minor reconstructions resistant identifier:

1. Identifier configuration contains all item names that shall be used to construct it. We use values of any item that exists in object instance record.
2. We serialize all item values found in previous step according to rules given by identifier configuration (plain order usually).

3. We extend serialized result with pre-processed (also serialized) description of full parent objects hierarchy.
4. We compute a hash from string created in previous step and make it temporary identifier.
5. We check for any duplicity after creating temporary identifiers for all records (all object instances) and make temporary identifiers the final ones when no conflict occurs. Otherwise we have to add SNMP index to serialized values and compute hash once again. But this should never happen when identifier set of items is properly configured.

In case of VCHANNEL object class identifier we decided to use the following items for the first implementation – some of them are imported from neighbor class according to SNMP indexes: *cpwVcType*, *cpwVcPsnType*, *cpwVcName*, *cpwVcDescr*, *ifKeyByDescrAndIP* (corresponding “local box” interface description including addressing – imported from G3 INTERFACE class), *cpwVcPeerAddrType*, *cpwVcPeerAddr*, *cpwVcRemoteIfString*.

There must also exist a textual – human readable description of virtual channel end-point. It is delivered from the same set of items. Here are some examples of real identifiers:

- ethernet over mpls, mplsNonTe, addVlan, EoMPLS tunnel from ACONET to PIONIER, TenGigabitEthernet4/2.403, Te4/2.403, EoMPLS tunnel from ACONET to PIONIER, peer=ipv4; 195.113.1xx.xx; EoMPLS Pionier-Aconet
- ethernet over mpls, mplsNonTe, addVlan, EoMPLS tunnel from SANET to PIONIER, TenGigabitEthernet4/2.402, Te4/2.402, EoMPLS tunnel from SANET to PIONIER, peer=ipv4; 195.113.1xx.xx; EoMPLS Pionier-Sanet
- ethernet over mpls, mplsNonTe, peer=ipv4; 195.113.1xx.xx
- ethernet over mpls, mplsNonTe, portBased, 1GE Lousiana – super-computing center, GigabitEthernet9/3, Gi9/3, 1GE Lousiana – supercomputing center, peer=ipv4; 195.113.1xx.xx
- ethernet over mpls, mplsNonTe, portBased, Cinegrid – 1GE to Prague, GigabitEthernet9/14, Gi9/14, Cinegrid – 1GE to Prague, peer=ipv4; 195.113.1xx.xx

Similar situation will occur in case of other vendor devices – we will just have to extend the set of items above with equivalent or similar ones from other vendor MIBs. This will not change any result for formerly defined set. The only thing we have to care for is to keep order for formerly configured items.

2.3 Items currently measured in VCHANNEL class

Currently implemented solution gives compact view on virtual channel end-points. It gives a lot of descriptive values and beside it what users expect – channel state information, byte and packet counters. An example of what is currently measured is shown in Figure 1. We didn't find any identifier collisions nor unstabilities nor measurement problems since september 2009 when we put this measurement into production mode.

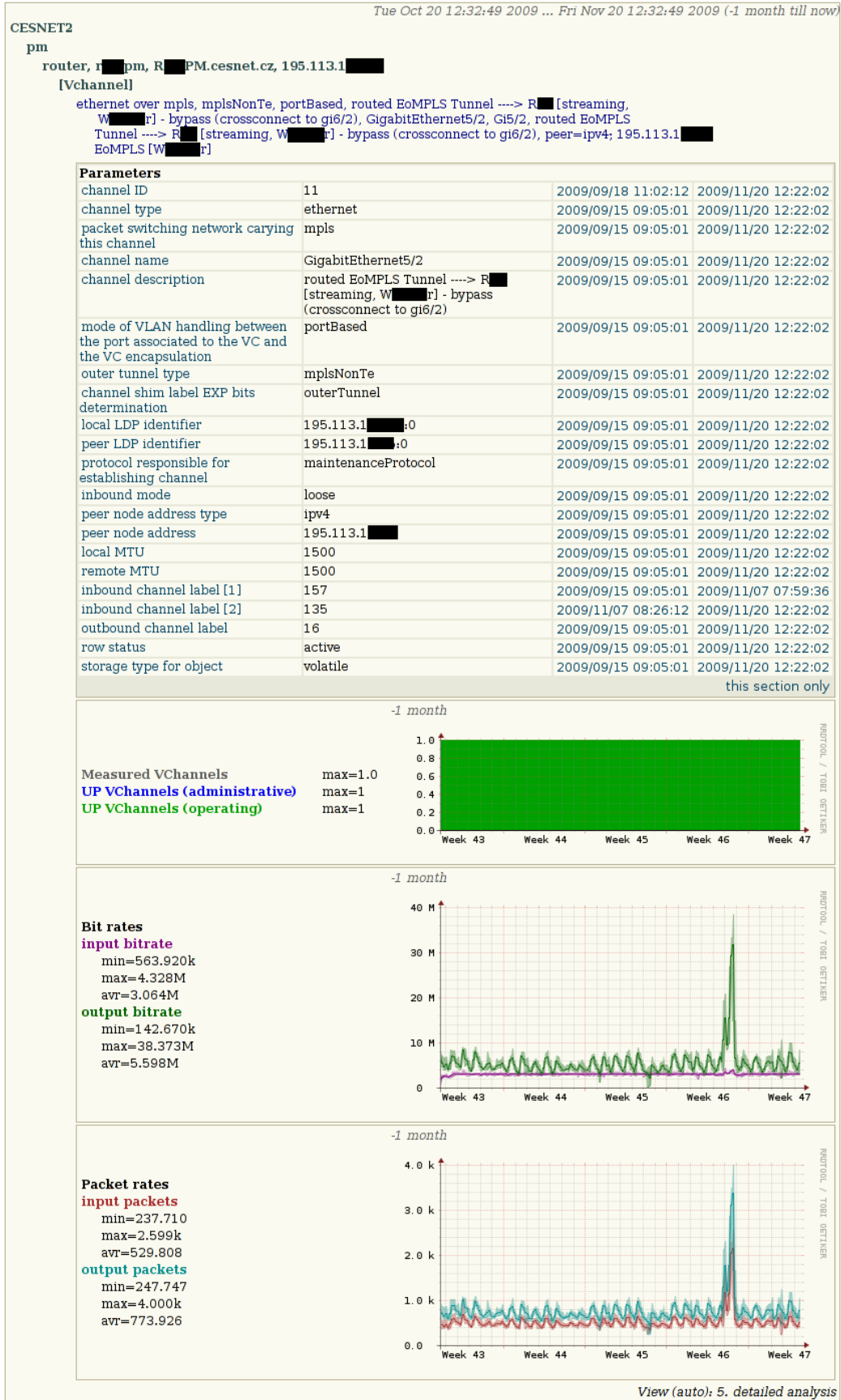


Figure 1. Vchannel example.

3 Class based QoS measurement

As was said above CESNET2 IP/MPLS backbone core is currently based on Cisco technology (CRS-1 and 76xx platforms). And also in this area we insist on keeping G3 system vendor and platform independent, so we have to solve the same problems as in virtual channels measurement. Quality of Services is not similar (at the logical structure level) to anything what is currently defined within G3 system. The typical logical tree has three layers with policy-maps at the top containing class-maps each consisting of sets of conditions (match-statements for example) and policies (real traffic regulation). However we decided to define new single object class QOS in G3 system for the first implementation and to solve internal QoS hierarchy with some analogy of object attributes.

3.1 MIBs used to measure Class based QoS in CESNET2 IP/MPLS backbone

The CISCO-CLASS-BASED-QOS-MIB is well documented MIB containing everything what is requested to be measured. The only thing which we wasn't able to resolve was (in our conditions with current OS versions on both platforms) proper ordering of objects.

3.2 G3 QoS identifiers

We wanted (as in previous case) to create identifier resistant to minor reconfigurations, SNMP index changes, system reboots and similar events that don't change the purpose and meaning of particular QoS object. As we said above the decision was to solve any parent-child relations with the help of attributes – so we have to create absolute identifier for each object at any layer of QoS logical hierarchy. Mechanism of identifier creation is the same as in the case of other G3 identifiers. It relies on defined set of items in appropriate order. There are the following items used (with relation to CISCO-CLASS-BASED-QOS-MIB) in the first implementations as sources of QOS identifier construction:

cbQosPolicyDirection, cbQosIfType, cbQosFrDLCI, cbQosAtmVPI, cbQosAtmVCI, cbQosEntityIndex, cbQosVlanIndex, cbQosObjectsType, cbQosPolicyMapName, cbQosPolicyMapDescr, cbQosCMName, cbQosCMDesc, cbQosCMInfo, cbQosMatchStmtName, cbQosMatchStmtInfo, cbQosPoliceCfgPir, cbQosPoliceCfgRate, cbQosPoliceCfgRate64, cbQosPoliceCfgBurstSize, cbQosPoliceCfgExtBurstSize, cbQosPoliceCfgPercentRateValue, cbQosPoliceCfgPercentPirValue, cbQosPoliceCfgCellRate, cbQosPoliceCfgCellPir, cbQosPoliceCfgBurstCell, cbQosPoliceCfgExtBurstCell, cbQosPoliceCfgBursiime, cbQosPoliceCfgExtBursiime, cbQosPoliceCfgCdot, cbQosPoliceCfgConformColor, cbQosPoliceCfgExceedColor, cbQosPoliceCfgConformAction, cbQosPoliceActionCfgConform, cbQosPoliceCfgConformSetValue, cbQosPoliceCfgExceedAction, cbQosPoliceActionCfgExceed, cbQosPoliceCfgExceedSetValue, cbQosPoliceCfgViolateAction, cbQosPoliceActionCfgViolate, cbQosPoliceCfgViolateSetValue

As we already said we are currently not able to reconstruct proper order of objects as is given by configuration (without downloading device configuration of course). It is the case of class-maps ordering within particular policy-map. None of indexes or measured OIDs from CISCO-CLASS-BASED-QOS-MIB gives the pos-

sibility to build it correct. Therefore we decided to prefix any class-map description with its parent policy-map description at the user interface level which virtually looks like the top QoS objects are class-maps. Here is an example how the QoS identifiers may look like at the visualization level (G3 UI navigation) – policy-map at particular interface:

- input policymap 'CESNET2-IP2MPLS-CE2PE-10GE-in' at TenGigabitEthernet1/2, Te1/2, Line 10Gbps cluster[classmap] 'CESNET2-RealTime-DSCP' (Real-Time) (matchAny)
 - [police] cir 40000000 bps, burst 625000 B, conditional burst 1250000 B, conform-action transmit, exceed-action drop, violate-action 0
 - [matchStatement] 'Match dscp cs5 (40) ef (46)'
- input policymap 'CESNET2-IP2MPLS-CE2PE-10GE-in' at TenGigabitEthernet1/2, Te1/2, Line 10Gbps cluster[classmap] 'CESNET2-NetworkControl-DSCP' (Network Control) (matchAny)
 - [police] cir 10000000 bps, burst 625000 B, conditional burst 1250000 B, conform-action transmit, exceed-action transmit, violate-action transmit
 - [matchStatement] 'Match dscp cs6 (48) cs7 (56)'
- input policymap 'CESNET2-IP2MPLS-CE2PE-10GE-in' at TenGigabitEthernet1/2, Te1/2, Line 10Gbps cluster[classmap] 'CESNET2-CriticalTraffic-DSCP' (Critical Traffic) (matchAny)
 - [police] cir 100000000 bps, burst 6250000 B, conditional burst 12500000 B, conform-action transmit, exceed-action transmit, violate-action transmit
 - [matchStatement] 'Match dscp cs2 (16) af21 (18) af22 (20) af23 (22) cs3 (24) af31 (26) af32 (28) af33 (30)'
- input policymap 'CESNET2-IP2MPLS-CE2PE-10GE-in' at TenGigabitEthernet1/2, Te1/2, Line 10Gbps cluster[classmap] 'CESNET2-BestEffort' (Best Effort) (matchAny)
 - [matchStatement] 'Match mpls experimental topmost 0 '
 - [matchStatement] 'Match dscp default (0)'
 - [set] marking position mplsExp
- input policymap 'CESNET2-IP2MPLS-CE2PE-10GE-in' at TenGigabitEthernet1/2, Te1/2, Line 10Gbps cluster[classmap] 'CESNET2-Video-DSCP' (Video) (matchAny)
 - [matchStatement] 'Match dscp cs4 (32) af41 (34) af42 (36) af43 (38)'
 - [police] cir 60000000 bps, burst 1875000 B, conditional burst 3750000 B, conform-action transmit, exceed-action transmit, violate-action transmit
- input policymap 'CESNET2-IP2MPLS-CE2PE-10GE-in' at TenGigabitEthernet1/2, Te1/2, Line 10Gbps cluster[classmap] 'CESNET2-LessEffort' (Less than Best Effort) (matchAny)
 - [matchStatement] 'Match dscp cs1 (8) af11 (10) af12 (12) af13 (14)'
 - [set] marking position mplsExp, MPLS value 1
 - [matchStatement] 'Match mpls experimental topmost 1 '
- input policymap 'CESNET2-IP2MPLS-CE2PE-10GE-in' at TenGigabitEthernet1/2, Te1/2, Line 10Gbps cluster[classmap] 'class-default' (matchAny)
 - [set] marking position mplsExp
 - [matchStatement] 'Match any '

3.3 Items currently measured in QOS class

There are basically three groups of information available – byte based, packet based and bit rate based. We prefer 64 bit counters in all cases of counter type values. Here is the list of OIDs (CISCO-CLASS-BASED-QOS-MIB) that are currently implemented:

Packet based OIDs

cbQosCMDropPkt64, cbQosCMNoBufDropPkt64, cbQosCMPrePolicyPkt64, cbQosMatchPrePolicyPkt64, cbQosPoliceConformedPkt64, cbQosPoliceExceededPkt64, cbQosPoliceViolatedPkt64, cbQosQueueingDiscardPkt64, cbQosSetAtmClpPkt64, cbQosSetDiscardClassPkt64, cbQosSetDscpPkt64, cbQosSetDscpTunnelPkt64, cbQosSetFrDePkt64, cbQosSetFrFecnBecnPkt64, cbQosSetL2CosPkt64, cbQosSetMplsExpImpositionPkt64, cbQosSetMplsExpTopMostPkt64, cbQosSetPrecedencePkt64, cbQosSetPrecedenceTunnelPkt64, cbQosSetQosGroupPkt64, cbQosSetSrpPriorityPkt64

Byte base OIDs

cbQosCMDropByte64, cbQosCMPostPolicyByte64, cbQosCMPrePolicyByte64, cbQosMatchPrePolicyByte64, cbQosPoliceConformedByte64, cbQosPoliceExceededByte64, cbQosPoliceViolatedByte64, cbQosQueueingCurrentQDepth, cbQosQueueingDiscardByte64, cbQosQueueingMaxQDepth

Bit rate based OIDs

cbQosCMDropBitrate, cbQosCMPostPolicyBitRate, cbQosMatchPrePolicyBitRate, cbQosPoliceConformedBitRate, cbQosPoliceExceededBitRate, cbQosPoliceViolatedBitRate

As you can see above there are some analogical items in byte based and bit rate based groups – both may represent the same traffic under some conditions. But the results may slightly differ in that case (as shown in given class-map statistics example – Figure 2). While byte based items represent average bit rate between two consecutive readings of particular counter, rate based item corresponds with average rate computed at the measured device – different timing makes different results. Here are examples for tree object types: class-map statistics in Figure 2, match-statement statistics example in Figure 3 and policy statistics example in Figure 4.

Current implementation of class based QoS measurement for mentioned platforms in G3 system is still rather experimental. It is a part of production version of the system, but is disabled by default (must be explicitly enabled for particular device, group of devices or the whole measurement) and we expect some tuning especially at the visualization level. However we didn't observe any impact on system stability and reliability.

4 G3 reporter extension

Initial implementation of G3 system reporter [1] was based on fixed sequence of tasks which were executed according to specific configuration. This scheme may satisfy almost all requirements in the area of single purpose reporting of specific set of objects (network cloud for example). But another situation is the case of multi-purpose (multi-output) reporting of the same set of objects in the same time periods where this scheme is far from being efficient – see Figure 5 as an example. Let's imagine that "Report A" may represent utilization of particular network

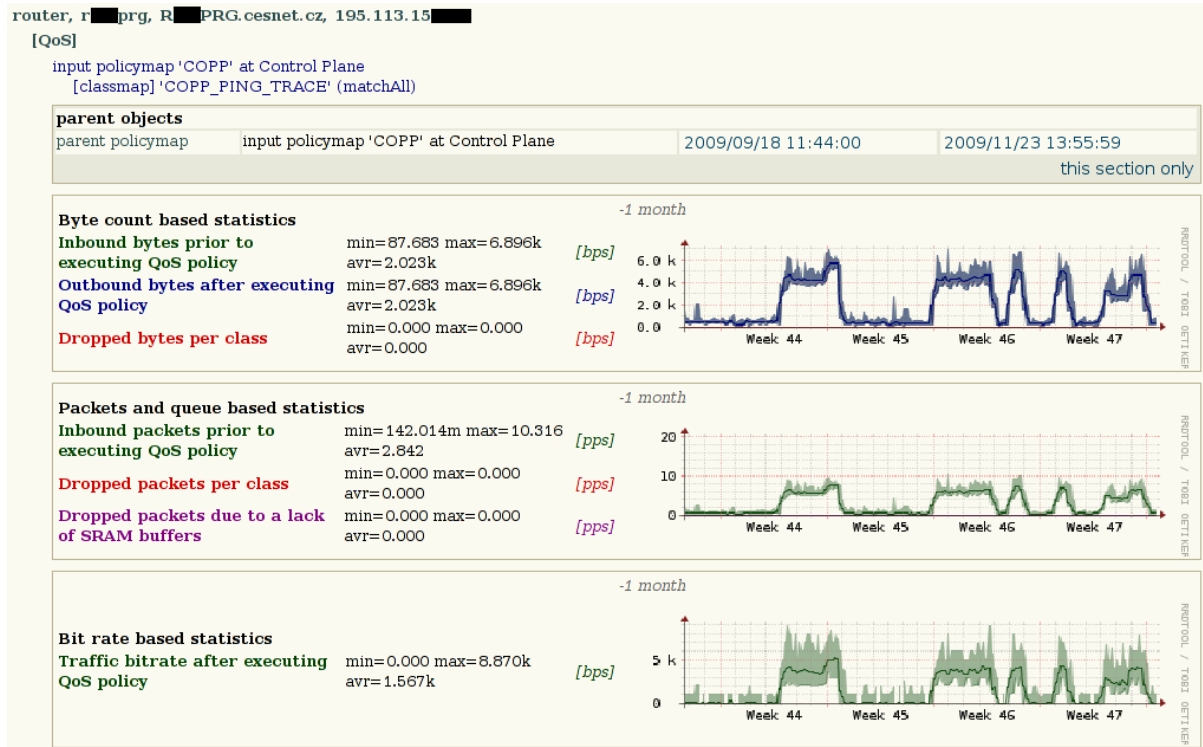


Figure 2. Class-map statistics example.

cloud and “Report B” error state characteristics of the same network cloud. The most demanding and thus slowest operations in G3 reporter processing scheme are performed for each report configuration despite the fact that some of the operations are identical (“Object identifiers preparation”). Other operations are similar (“Statistics computing”), but they are performed over the same set of prepared identifiers and could be efficiently (fast operation) reused during period when are valid (retrieved identifiers expire according to configuration). In independent processing they have to be periodically rediscovered (slow operation) for each report separately. It may not be a problem when only a few objects are configured to be processed and when requested time periods are relaxed in relation to number of operations requested. But when we have to process hundreds of objects in parallel mode we may get into troubles.

One of our motivations to extend the processing scheme and set of configuration options to achieve more efficient processing of multi-purpose reports was IP/MPLS backbone of CESNET2 network (NREN in the Czech Republic) reporting in utilization and error state perspectives. Another example is monitoring of core network infrastructure of the FEDERICA project (European community cofunded project in the FP7 in the area “Capacities – Research Infrastructures” – <http://www.fp7-federica.eu/>) which is one of the tasks we are responsible for. For example FEDERICA core network infrastructure consists of hundreds of interconnections and independent processing of “network utilization” and “network health” reports demands a lot of additional resources.

To make things faster and efficient we decided to extend G3 processing architecture in the first step. As shown in Figure 6 we enabled to split up data preparation

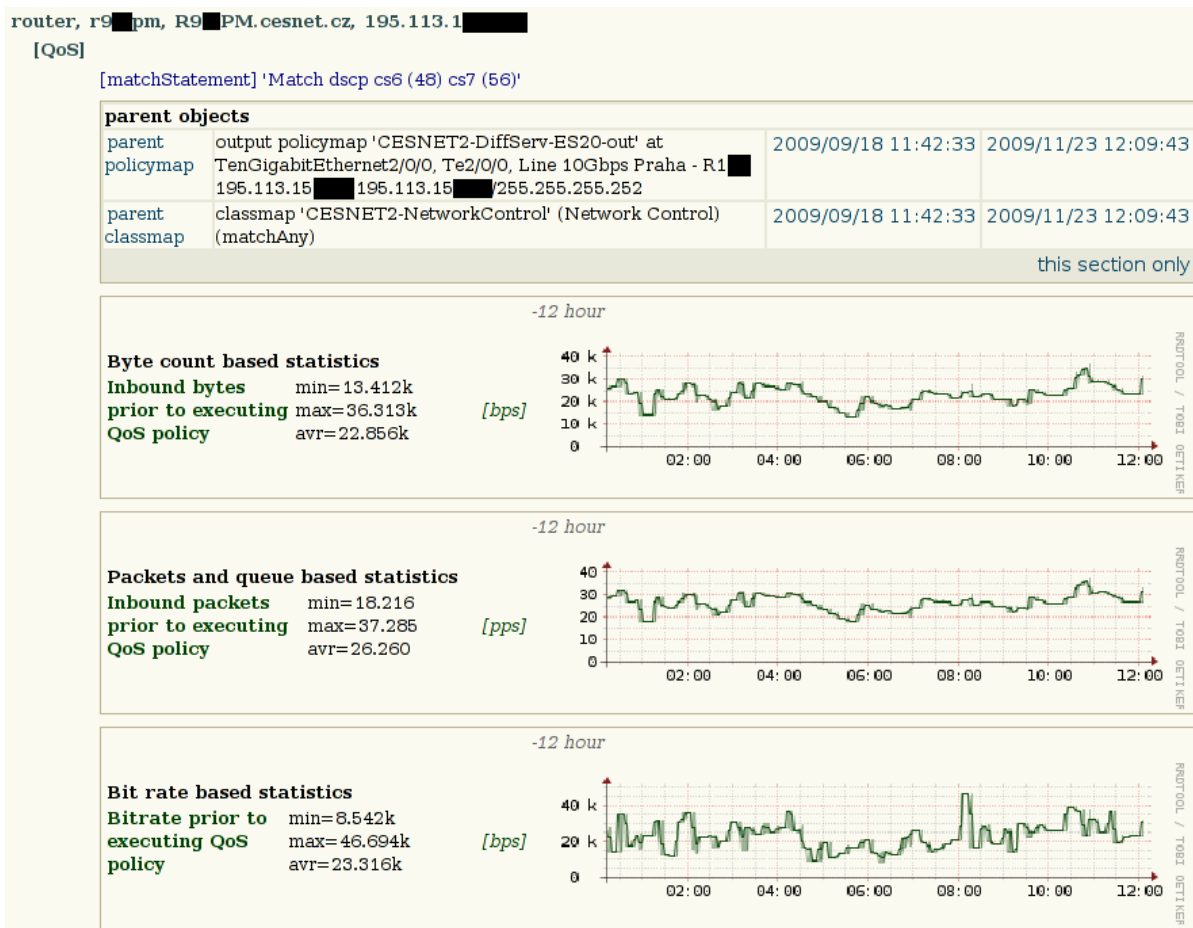


Figure 3. Match-statement statistics example.

processing part from the part that performs visualization.

New architecture implies periodical import of data prepared by the data processing part (left) into the visualization part (right). To make it efficient we need to change the frequency of data updates and imports. In original processing scheme any specific operation (utilization computing for example) was executed over all objects for which particular information (utilization value for example) expired in a continuous run. It means that specific operation could be executed on 100% of objects in a single step.

Therefore we implemented configurable limits for any specific operation to reduce number of objects that will be processed in single step. We also changed working data savings scheme – computed data are flushed to storage after finishing every computing step now. This results in frequent updates of working data in the data processing part – we can set limit to 8 objects for example for which statistical data are computed in a single step and then immediately store the results. We got also better distribution of processing in the time line for both parts. Also in case of visualization part we may configure for example limit of 4 detailed reports to be processed in a single step and then optional update of summary report will start immediately.

Last but not least we changed computed objects ordering strategy. Especially in network environment we are interested in information about both directions of

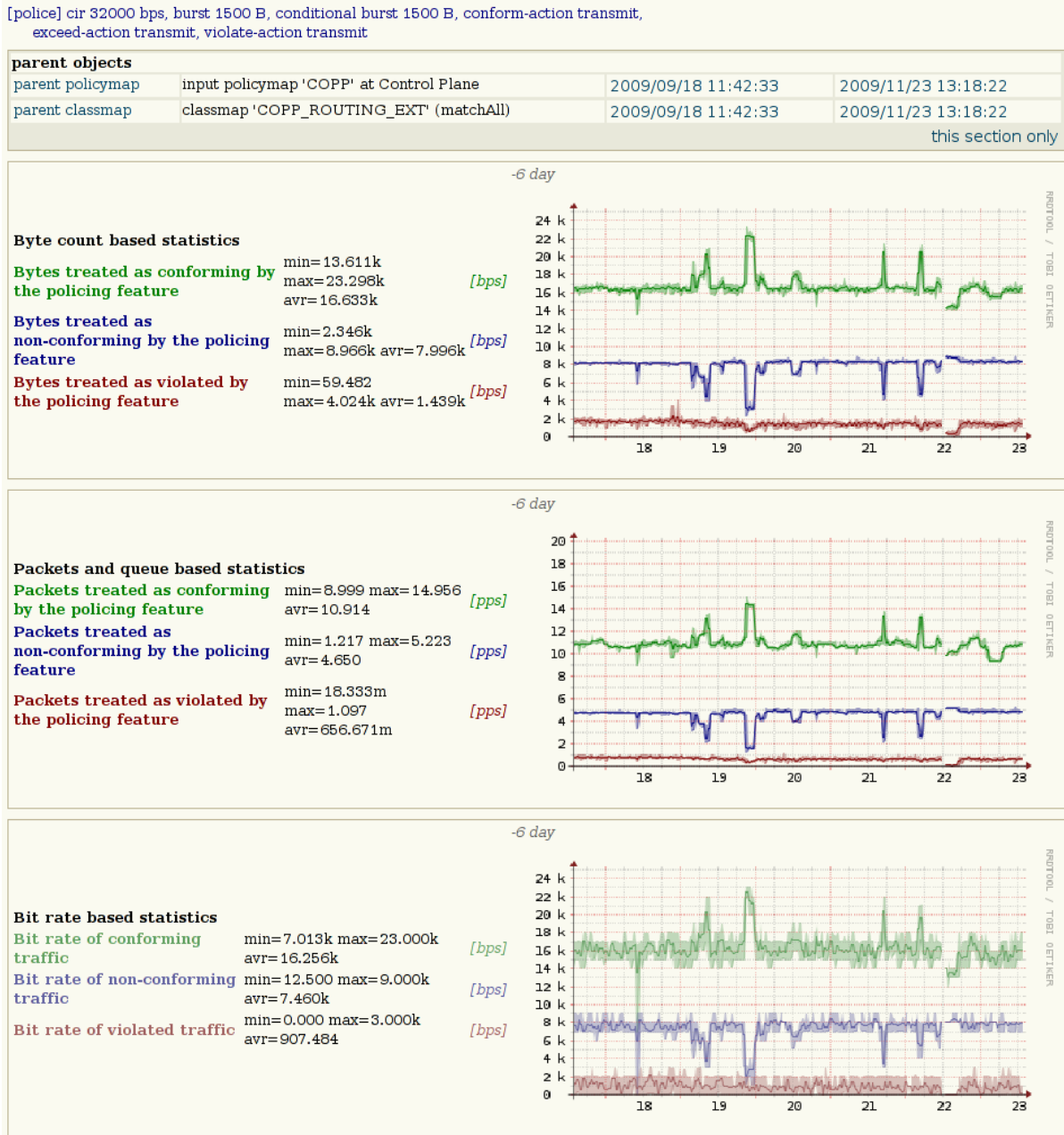


Figure 4. Police statistics example.

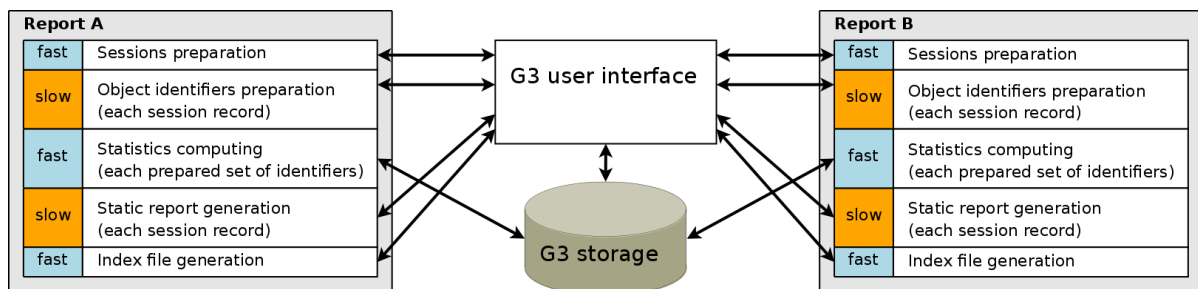


Figure 5. Standard G3 reporter architecture in parallel processing.

data communication on a single line. But there is a lot of cases when we have to

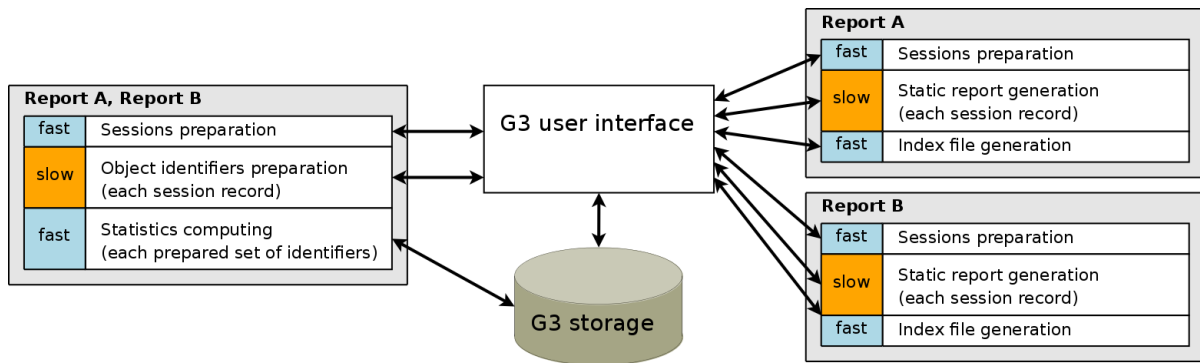


Figure 6. Extended G3 reporter architecture supporting multi-purpose reports.

compute results for both directions from data corresponding with one end point of the line only. It is in principle independent computing of each direction inside the system but with the same object identifiers. And as was said above these interface identifiers are valid during limited time period only. When shall be used twice (for each traffic direction) without expiration we may have to reorder objects in that way. Another reason for reordering objects is request for executing several specific operations (line utilization, interface errors) for each object – also here we want to run all specific operations (even those which former results didn't expired yet) one after another for single object while its identifiers are valid (it is several times faster than object identifiers preparation). Therefore we implemented option that enables forced objects ordering (in each processing step) that is given by the first specific operation and inherited by the others.

All these major extensions together with other minor code optimization caused approximately four times faster processing (in average) of multi-purpose reports (at least two types of outputs over the same set of objects) in comparison with former processing scheme. That enables to shorten the time period of generated reports validity on one side and decreases bursts of system load on the other side. As an example where this approach brings valuable benefit we present outputs of both cases mentioned above – partial snapshot of the new series of CESNET2 IP/MPLS backbone maps Figure 7 and part of “network health” summary report in Figure 8 with corresponding detailed report for particular interface in Figure 9 both from FEDERICA network core infrastructure monitoring.

4.1 Acknowledgement

This work is supported by the research intent MSM6383917201.

References

- [1] KOŠŇAR, T. *G3 System – Reporter* Technical Report 14/2007¹, CESNET: Praha, 2007.

¹ <http://www.cesnet.cz/doc/techzpravy/2007/g3-reporter/>



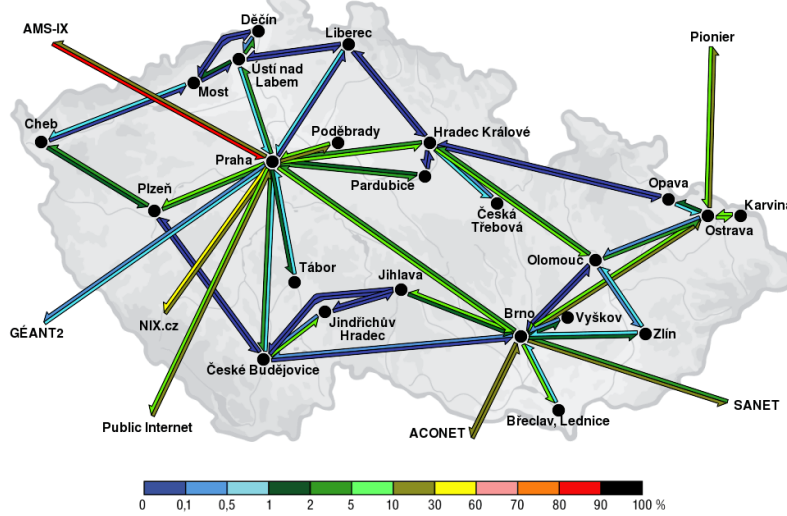
CESNET2: IP/MPLS backbone utilization

Average load for period: 2009/11/30 11:01:01 - 2009/11/30 11:11:01 SWT (Europe/Prague), GMT+1H

Průměrné zatížení v období: 2009/11/30 11:01:01 - 2009/11/30 11:11:01

The map shows the actual load of the IP/MPLS backbone in the CESNET2 network. The presented links are coloured by average load. Click on the line to obtain detailed statistics.

Mapa znázorňuje aktuální zatížení IP/MPLS páteře sítě CESNET2. Zobrazené linky jsou zbarveny podle průměrného zatížení. Klepnutím na linku získáte podrobnější údaje.



Lines

The following table shows values of other relevant parameters measured at active network devices during the period: 2009/11/30 11:01:01 - 2009/11/30 11:11:01 SWT (Europe/Prague), GMT+1H.

V následující tabulce jsou k dispozici další související údaje naměřené z aktivních prvků sítě za období 2009/11/30 11:01:01 - 2009/11/30 11:11:01.

Link	Utilization (avr)	Capacity	Bitrate (avr)
ACONET->Brno	18.045 %	10.000 Gbps	1.804 Gbps
AMS-IX->Praha	81.116 %	1000.000 Mbps	811.161 Mbps
Břeclav, Lednice->Brno	0.586 %	300.000 Mbps	1.758 Mbps
Brno->ACONET	15.772 %	10.000 Gbps	1.577 Gbps
Brno->Břeclav, Lednice	6.634 %	300.000 Mbps	19.902 Mbps
Brno->České Budějovice	0.292 %	20.000 Gbps	58.401 Mbps
Brno->Jihlava	6.169 %	2.000 Gbps	123.379 Mbps

Figure 7. "Utilization map" of CESNET2 IP/MPLS backbone.

FEDERICA: network infrastructure health

Peaks of potential problem rates for period: 2009/11/26 14:22:24 - 2009/11/26 14:32:24 SWT (Europe/Prague), GMT+1H

The map below shows the actual overall health of the FEDERICA project core network infrastructure. The presented links are coloured by peaks of potential problem rates. Click on the line to obtain detailed link report. Detailed link reports are currently generated every 3600 seconds for period[s] starting at '-24 hours,-7 days,-6 months' and finishing 'now'.

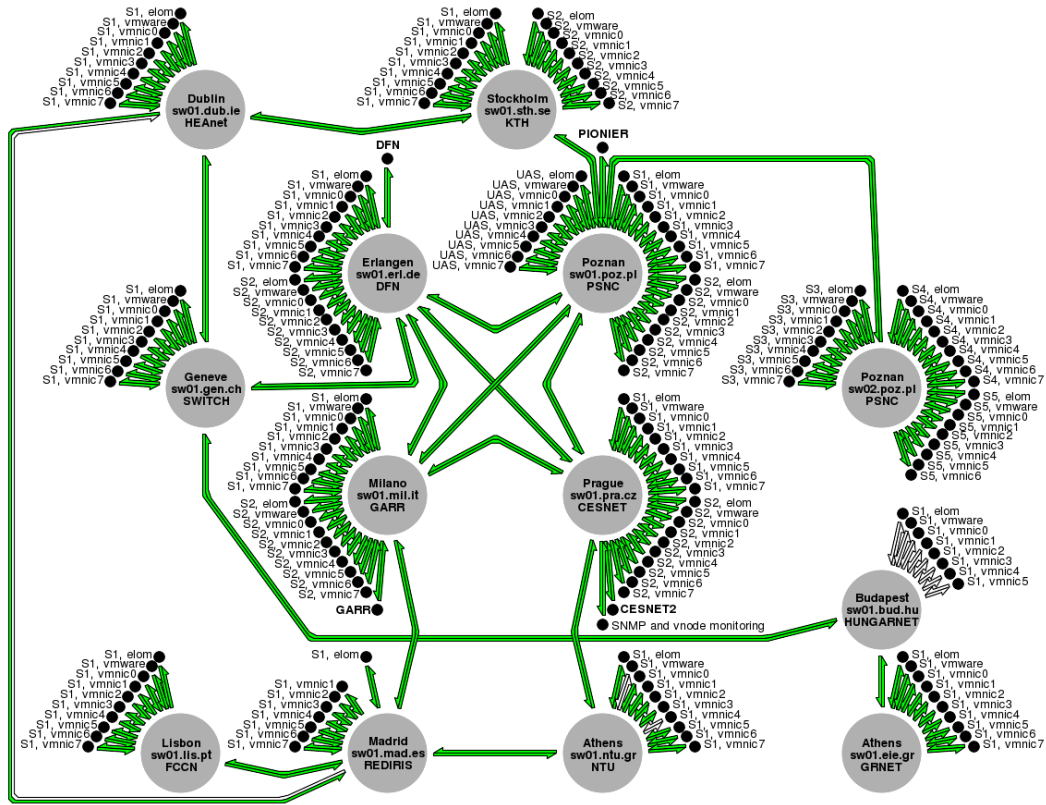


Figure 8. “Health map” of FEDERICA network.

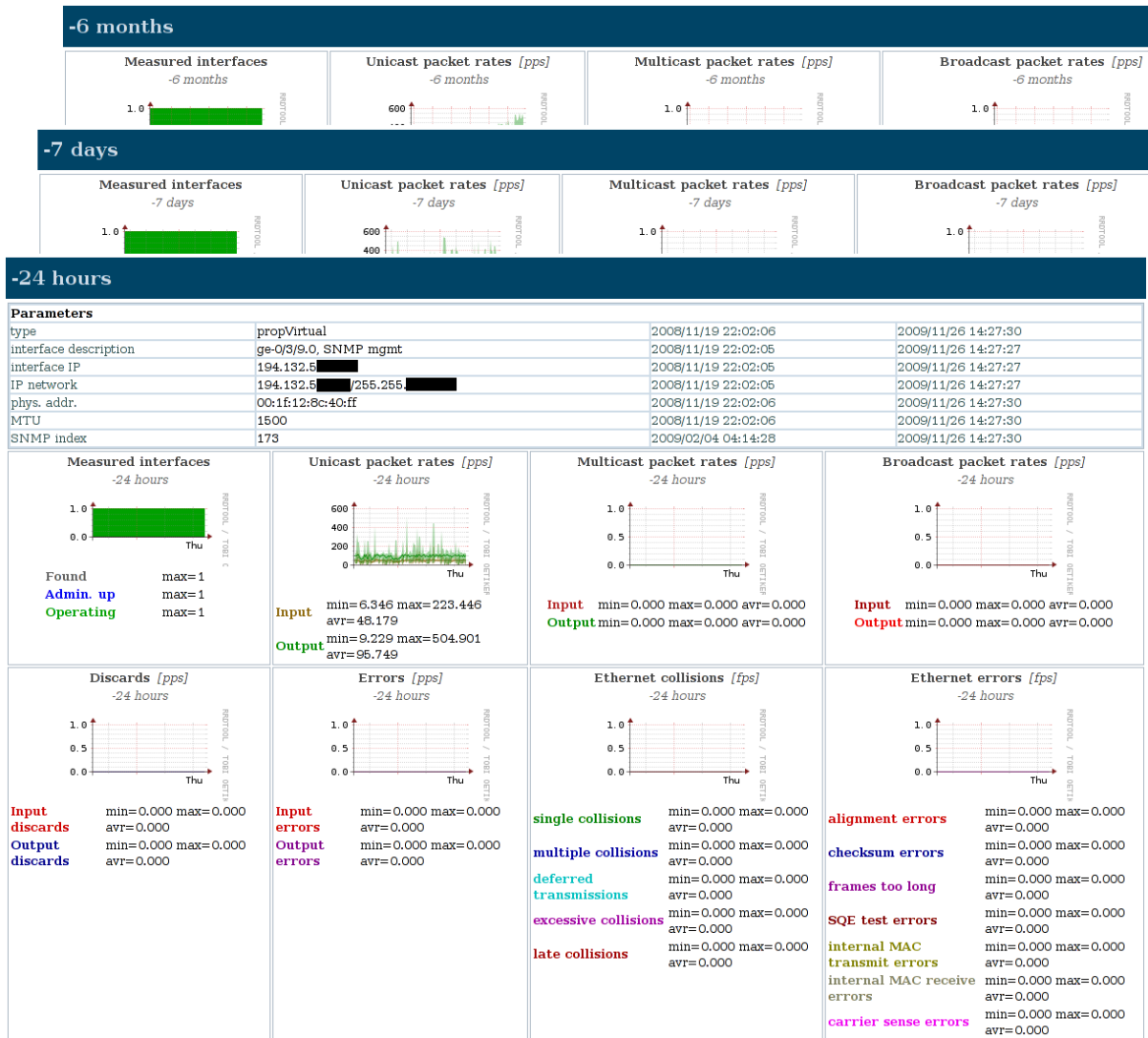


Figure 9. "Health view" on particular interface (picture modified) in FEDERICA network.