Networking Studies III

Selected TechnicalReports

CESNET 2009
All CESNET technical reports are available at
Preface

Since 2004, CESNET has been working on a research plan “Optical National Research Network and its New Application”. Execution of this plan is scheduled for seven years (2004–2010) and the research activities are funded by the Czech Ministry of Education, Youth and Sports and CESNET Association members.

We must point out that we regard the assurance of the research plan activities financing for a seven-year period as a great success. This period is reasonably long for starting and successfully completing extensive research activities. We are currently negotiating with the Ministry of Education, Youth and Sports about a future funding model. We believe these negotiations will be successful and appropriate funding will be provided for the period between 2011 and 2015.

Considering the enormous range of research work and large number of researchers participating in this work, we divided the research plan work into ten thematic activities:

- CESNET2 Backbone Network Development
- Optical Networks
- Programmable Hardware
- Network and Traffic Monitoring
- Performance Monitoring and Optimization
- AAI and Mobility
- METACentre
- Virtual Collaborative Environments
- CESNET CSIRT
- Support of Applications

The activities comprise areas from the lowest transmission layers through middleware, authentication, authorization and security up to research and development of application services. Each activity has its head and a deputy head who coordinate a particular team and are responsible for professional level and effective utilization of allocated funds. The results achieved in framework of the activities are internally evaluated within the CESNET Association twice a year and the results of the evaluation are used to improve the research efficiency in the successive periods. We put a lot of effort into improving the collaboration and interaction of activities and obtaining user feedback. Our goal is to provide the users with the results of the research activities as soon as possible.

A very important aspect of our work is the participation in international research projects. In addition to joining “traditional” GNx and EGEEEx projects,
we plan to extend our involvement in the framework of Future of the Internet projects initiated and supported by the European Commission. Together with other key players in our country – governmental agencies, universities, industry, ISP community and others – we plan to launch The Czech Future Internet Forum.

In addition to publishing research results in scientific journals and presenting them on conferences, three years ago we also started to publish last year's research results in the form of selected technical reports. The members of the CESNET Research Plan Steering Board selected eleven reports from the total of twenty-five that were published in 2008. I believe you will find this form of informing about the results of our work interesting and also hope it will contribute to a more intensive international cooperation in the very attractive and fast-growing area of computer networking and its applications.

To conclude I would like to thank all my colleagues working on research plan for their exceptional efforts put to their work. Let me also use this opportunity and thank all the funding bodies – without their contributions our work wouldn't be possible.

Jan Gruntorád
Member of the Board of Directors and CEO of CESNET
# Table of Contents

**Part I: CESNET2 Network**

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CESNET2 IP/MPLS Backbone Network Design and Deployment in 2008</td>
<td>3</td>
</tr>
<tr>
<td>Václav Novák, Petr Adamec, Pavel Šmrha, Josef Verich</td>
<td></td>
</tr>
<tr>
<td>Open Photonics DWDM Systems Deployment within the CESNET2 Network and Single Fibre Bidirectional Operation</td>
<td>19</td>
</tr>
<tr>
<td>Josef Vojtěch, Václav Novák, Miloslav Hůla</td>
<td></td>
</tr>
<tr>
<td>QoS Design and Implementation in the CESNET2 MPLS-based Backbone</td>
<td>29</td>
</tr>
<tr>
<td>Pavel Šmrha, Josef Verich</td>
<td></td>
</tr>
</tbody>
</table>

**Part II: Network Monitoring and Security**

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is Spam Visible in Flow-Level Statistics?</td>
<td>67</td>
</tr>
<tr>
<td>Martin Žádník, Zbyněk Michlovský</td>
<td></td>
</tr>
<tr>
<td>G3 System – Distributed Measurement Architecture</td>
<td>79</td>
</tr>
<tr>
<td>Tom Košňar</td>
<td></td>
</tr>
<tr>
<td>Designing a Hardware-Accelerated Firewall with Two 10 Gbps Ports</td>
<td>101</td>
</tr>
<tr>
<td>Viktor Puš, Tomáš Dedek</td>
<td></td>
</tr>
<tr>
<td>Impact of Network Security on Speech Quality</td>
<td>113</td>
</tr>
<tr>
<td>Miroslav Vozňák</td>
<td></td>
</tr>
</tbody>
</table>

**Part III: Middleware and Applications**

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-Stamp Authority</td>
<td>127</td>
</tr>
<tr>
<td>Vladimír Smotlacha, Milan Sova</td>
<td></td>
</tr>
<tr>
<td>eduroam Authentication over Jammed Network</td>
<td>135</td>
</tr>
<tr>
<td>Jan Tomášek, Milan Sova</td>
<td></td>
</tr>
<tr>
<td>Implementation of DXT Compression for UltraGrid</td>
<td>149</td>
</tr>
<tr>
<td>Ian Wesley-Smith, Miloš Liška, Petr Holub</td>
<td></td>
</tr>
<tr>
<td>CAVE to CAVE: Communication in a Distributed Virtual Environment</td>
<td>161</td>
</tr>
<tr>
<td>Roman Berka, Zdeněk Trávníček, Vlastimil Havran, Jiří Bittner, Jiří Žára, Pavel Slavík, Jiří Navrátil</td>
<td></td>
</tr>
</tbody>
</table>
Part I

CESNET2 Network
CESNET2 IP/MPLS Backbone Network
Design and Deployment in 2008

Václav Novák, Petr Adamec, Pavel Šmrha, Josef Verich

Abstract
This paper describes recent changes and innovations of the CESNET2 IP/MPLS network layer in 2008. The article is divided into the following logical parts: Current Network Design, Planned Network Design and CRS-1/16 Implementation, Prague PoP Splitting, CESNET2 Network Resilience and Availability. The Current Network Design part deals with the actual IP/MPLS layer and the underlying DWDM layer topologies based on the Cisco 7600 and newly introduced Cisco CRS-1 core routers as well as the Cisco ONS 15454 MSTP DWDM platform. The Planned Network Design and CRS-1/16 Implementation part covers the deployment of the new advanced “carrier-class” Cisco CRS-1 core router using the Secure Domain Router (SDR) technology in the Praha PoP. The Praha PoP Splitting part describes the first phase of the planned physical splitting of the currently single Praha PoP into two geographically distributed PoPs Praha I. and Praha II. to ensure even higher network core resilience through further redundancy. The CESNET2 Network Resilience and Availability part covers several techniques for achieving high availability of network services such as introduction of new backup circuits as well as advanced protocol and configuration features including BFD (Bidirectional Forwarding Detection), NSF (Non-Stop Forwarding) with the SSO (Stateful Switch-Over) to achieve very rapid (sub-second) convergence in case of link or module failures.

Keywords: CESNET2, optical network, DWDM, MPLS, CzechLight

1 Current CESNET2 Network Design

The CESNET Association operates a national DWDM and IP/MPLS network (CESNET2) covering multiple regions in the Czech Republic. The current CESNET2 network is built around a DWDM core providing 10 Gbps channels. It is based on Cisco ONS15454 MSTP system with the ROADMA technology. The main DWDM ring interconnects PoPs Praha – Brno – Olomouc – Hradec Králové – Praha with a transmission capacity of up to 32 optical channels in C-band at a speed up to 10 Gbps. The ONS 15454 MSTP based DWDM system is complemented by another system based on the family of open photonic devices called CzechLight (CL) that were developed by CESNET [3]. The key motivation for CL DWDM systems is a provision of a cost efficient solution for the delivery/provision
of gigabit (1-10 Gbps) connectivity to small PoPs and creation of backup connectivity for any PoPs based on pure optical circuits. A NIL (Nothing-in-Line) approach, i.e. optical line design without the inline components (e.g. amplifiers and chromatic dispersion compensators) was used.

The current CESNET2 optical topology is shown in Figure 1.

The IP/MPLS layer of the CESNET2 network follows the optical transmission topology. In the DWDM nodes of the main optical ring (Prague, Brno, Olomouc and Hradec Králové), CESNET operates backbone IP/MPLS routers as the P node elements within the MPLS network topology. In major PoPs, there reside the access routers functioning as PE routers which provide all the functionality and services of the backbone network (MPLS, EoMPLS, QoS, IPv4/IPv6 unicast, IPv4 multicast routing and NetFlow statistics). Both the P and PE routers are Cisco OSR 7609’s with the SUP720-3BXL processors and 1 GE and 10 GE line cards.

Small PoPs without MPLS functionality run L2/L3 switches (Catalyst 3750G) functioning as/representing CE devices from the MPLS perspective. The parent
PE routers achieve the full network services capability. There are trunks between PE and CE devices and configured VLANs. For the EoMPLS (Ethernet over MPLS) L2 services extension to the CE PoPs, CESNET2 uses EoMPLS tunnels-mapping into local VLANs.

The detailed IP/MPLS topology is shown in Figure 2. There are MPLS P routers in the core (marked as red) and PE routers (marked as blue). CESNET runs OSPFv2/OSPFv3 as an IGP routing protocol in the MPLS core and iBGP between the PE routers with the route reflectors on the Internet peering routers R84, R85 and R98. The same route reflectors are used for iMBGP (interior Multi-protocol BGP) to exchange routing information for IPv4 multicast as well as for IPv6 unicast.

CESNET2 supports hybrid unicast IPv4/IPv6 transmission using PE/6PE technology (dual-stack mode). The IPv4 unicast and multicast topologies are not congruent (unicast IPv4/IPv6 packets are MPLS-switched whereas multicast packets are transported at the IPv4/IPv6 layer).

Figure 2. CESNET2 IP/MPLS topology at the beginning of 2008

The deployment of the core P and PE routers, Cisco 7609, as the main routing
platform started in 2002. For current hardware and software features virtually all components have been gradually upgraded (chassis, CPU, interface cards). In the current configuration, routers are typically equipped with 10 GE interface cards with the throughput up to 40 Gbps/slot. With the demands on the properties and functions of the network and the increasing volume of traffic a number of problems arose, especially with the main Internet peering routers, such as high CPU load and thus longer BGP convergence, TCAM table overflow in the case of NetFlow v9 data export, and the lack of TCP flags in NetFlow v9 records (which is a hardware limitation of the SUP720-3BXL or RSP720-3CXL CPUs). All these problems can negatively influence backbone network performance and stability in the future. We have the following network research plans: to test and deploy new technologies, such as 40 and 100 Gbps, IPoDWDM principles, integration with the DWDM network, and other new advanced features. Therefore, we started the selection process of new and more powerful advanced routers for the main core of the network, which will enable further smooth development of CESNET2. The basic requirements were as follows:

- A modular “carrier class” router with a life span of at least 5-7 years
- Fully redundant system HW components (PS, fans, CPUs, switching fabric)
- In-service upgrade to higher backplane/switching and interface capacities (100 Gbps) without the traffic forwarding losses and outages
- Non-blocking switching matrix architecture; IPv4/IPv6 unicast and multicast switching in HW
- Switching performance of at least 700 million pps for IPv4 and IPv6 protocols
- HW and SW upgrades not affecting service, hot-swap for all components
- A modular operating system supporting per process upgrade or restart without forwarded packet losses or other negative impacts on service
- Quality of Service features support based on L2/L3/L4 attributes (DiffServ according to RFC2474, RFC2475, RFC2597, RFC2598, RFC2697, RFC3270)
- MPLS capabilities and derived services (EoMPLS, VPLS)
- Traffic recording and reporting capability
- Management and maintenance capability
- CLI/SSH2/Telnet/SNMPv3, NetFlow v9 management
- IPv4/IPv6 unicast and multicast features including protocols OSPFv2/v3, ISIS, BGP, MPLS VPN, EoMPLS, MPLS TE, PIM, IGMP, MSDP, BFD, ...
- 1 and 10 Gbps interfaces, 40 Gbps interfaces available (OC-768), 10GE Tunable WDM PHY
- Separated data and control planes, control plane security features
- Non-stop forwarding for OSPF, BGP, LDP and RSVP
— Logical or virtual routers support
— Multi-shelf system support
— Management of configurations with the possibility of a return to previous versions (configuration rollback) including the option of editing, verification of the functionality of configuration and its confirmation
— Full availability and stability of all CESNET2 services (IPv4 unicast / multicast, EoMPLS) and full compatibility with the CESNET2 network, including DWDM optical transmission system based on ONS 15454-MSTP technology

In the first phase of the migration process we planned new router(s) installation in the Praha PoP. The current Internet peering and core routers R84 and R107 will be replaced (see Figure 2). The final solution could be based on two physical routers or one router with logical router support to configure the required PE and P functionality. The next phases will include a new router deployment in the Brno PoP and a 40 Gbps DWDM line Praha-Brno installation (over the current DWDM system). The expected final result with the planned physical Praha PoP splitting is depicted in Figure 3.

Both new PE and P routers must be connected by at least 20 Gbps capacity. Considering the solution based on the logical routers (i.e. within one physical router), we require that control plane and dedicated HW and SW resources for each PE an P router must be separated.

The public tender winner was the solution based on a Cisco CRS-1/16 router and logical PE and P routers configuration. Summary of the key features and characteristics of quoted CRS-1/16 configuration are:

— Secure Domain Routers (SDRs) dividing the single physical system into multiple logically separated routers. SDRs are isolated from each other in terms of their performance, availability and resources (e.g. memory + processors are not shared and CPU resources are not in contention). Some resources, such as chassis control, cooling, power, switch fabric, and partitioning, are shared with the rest of the system. The target CRS-1 setup is based on two SDRs replacing the Cisco 7609 devices in Praha: R84 (PE/node) and R107 (P-node)
— PE and P interconnected by 40 Gbps (OC-768 POS)
— Tunable DWDM 10GE interfaces with E-FEC/FEC/No-FEC (50 GHz spacing, C-band, 80 channels)
— IPv4/IPv6 unicast and multicast features (IPv4/IPv6 dual-stack) including protocols OSPFv2/v3, ISIS, BGP, MPLS VPN, EoMPLS, MPLS TE, PIM, IGMP, MSDP, BFD, ...
— Full QoS support
— Modular IOS-XR operating system, ISSU (In Service Software Upgrade)
Figure 3. New core routers planned implementation

- Full redundancy (1:1 for PS, fans, RPs, DRPs, control and alarm modules; 1:8 for switching fabric)
- Proposed HW configuration for PE and P nodes with no over-subscription
- Switching fabric performance of 40 Gbps full-duplex per slot, ready for in-service upgrade to 100 Gbps
- 4 free slots for future extension, multi-chassis support
- CLI/SSH2/Telnet/SNMPv3, NetFlow v9 management
- Full interoperability with the current IP/MPLS and DWDM ONS15454 optical transport infrastructure
- 40 Gbps WDMPOS interface module available

More information about this hardware and its features is available on the
In 2008 we also focused on CESNET2 IP/MPLS network features and services improvement. The main areas of our interest were network resilience and availability, IPv6 multicast and new QoS design and deployment.

The initial phase of QoS implementation in the CESNET2 IP/MPLS network was completed in 2005. The second network QoS deployment phase in the CESNET2 MPLS backbone was accomplished in 2008. It covered mainly service class restructuring, adding IPv6 support and unification of the QoS configuration for new hardware platforms or modules (e.g. CRS-1 and Cisco 7600 ES20 modules). The basic goal of Quality of Service (QoS) is to provide service differentiation among packets in the network. Such a service differentiation is noticeable during periods of network congestion (i.e. in case of contention for resources) and results in different levels of network performance. The technical approach was to implement DiffServ within the MPLS-based CESNET2 backbone using the E-LSP point-to-cloud (destination unaware) QoS model for 6 classes of traffic aggregates with different QoS characteristics. To ensure the DSCP transparency for transported IPv4/IPv6 user traffic the so called “Short-Pipe” MPLS DiffServ tunnelling mode was deployed. The QoS implementation for IPv4/IPv6 traffic currently uses 6 classes of service: Real-Time (EF-PHB), Network Control (AF-PHB), Video (AF-PHB), Critical Traffic (AF-PHB), Best Effort (default PHB), and Less than Best Effort (AF-PHB). The QoS admission control has been applied by means of policing at all ingress of the CESNET2 MPLS DiffServ and is available for academic user traffic. The detailed QoS design/redesign is described in detail in the technical report [4].

2 Planned Network Design and CRS-1/16 Implementation

In order to increase the capacity and stability in the current CESNET2 network we implemented the CRS-1/16 routing platform in the core Praha PoP. Performance upgrade was the main driver for implementing CRS-1 providing 40 Gbps/slot capacity allowing CESNET2 to deploy 40Gbps interfaces (OC768/STM256). The second objective was to simplify the architecture of the Praha I PoP and reduce operational costs (with fewer devices to manage). This could be achieved by consolidating the edge and peering functionality on a single CRS-1/16-Slot node and implementing Secure Domain Routers.

The migration process was divided into the following phases:

— Physical CRS-1/16 installation (Cisco partner ICS)

The CRS-1/16 “carrier-class” routing platform has specific requirements for installation and environment compared to traditional routers. The chassis size is DxWxH 60x100x213cm, its weight is approx. 700 kg (with card included), max. power of 11 kW, and cooling of about 32000 BTU/hr. The installed HW configuration included newest MSC card with lower power requirements, so the real CRS-1/16 power consumption is just about 5.5 kW. After powering up, the system boots for about 30 min (diagnostic tests, IOS-XR loading into all installed cards), so a very stable UPS is necessary. This is the basic design concept of CRS-1/16, because all the maintenance works (HW and SW upgrades) can be done in service.

The physical installation included OOB equipment for all main and backup RPs (Route Processor) and DRPs (Distributed Route Processor).

The CRS-1 power system is fully redundant and consists of two AC power shelves, each with three AC rectifiers, and an in-shelf alarm module. The chassis power distribution system divides the chassis into physical ‘load-zones’. Whilst the power system is designed to provide resilience in the event of a failure, a double
failure may result in critical line cards losing power and hence in the node being cut off from the rest of the network. With respect to the physical network design and topology, it is very important to plan slot allocation for high reliability. Three factors have been considered:

- Even distribution between power zones.
- A mixture of core- and edge-facing links within a power zone
- Distributing redundant DRPs between power zones

The final allocation of PE-SDR, P-SDR and LC (Line Card) and power zone selection is shown in Figure 5.

The chassis is populated from the periphery to the centre starting with slots 0, 1, 6, 7 in the upper PLIM Card Cage assigned to PE-SDR and slots 8, 9, 10, 13, 14 and 15 in the lower PLIM Card Cage assigned to P-SDR. Slots 3 and 4 in the upper PLIM Card Cage are reserved for PE-SDR DRPs and will not be used for line cards. As noted above, if one of the shelves or one of the rectifiers were to fail, there is sufficient capacity to ensure that components within each of the load-zones will continue to operate. However, to provide redundancy for a double failure of respective A and B rectifiers, the line cards will be spread across the different power zones, taking into account physical link redundancy and consistent slot placement policies across all chassis. The slot placement will ensure that at least one DRP and at least one SIP800 providing 10GE connections to redundant core sites will be operational under a “double rectifier failure” condition.

The IOS to IOS-XR Configuration Conversion process covered migration of R84 and R107 IOS configurations, adjusting the configurations according to IOS-XR specifics (Flexible NetFlow CLI, BGP RPL) and some new features in IOS-XR (LDP NSR, MPP) implementation. In the migration process specific CRS-1/16 features have been tested and verified before implementation (i.e. IPv4/IPv6 unicast/multicast, BGP/OSPFv2/OSPFv3 protocols, NetFlow v9, MPLS and overall network compatibility and stability, and many others). No recent problems have been found, so the migration process successfully finished without any impact on CESNET2 network stability and operation. We experienced the following issues:

- CRS-1/16 HW does not support low-speed 100BASE-TX interface. An external switch with the 1 Gbps uplink was installed to enable attachment of the required connections from the R84 router.
- MTU size of 9216 bytes on CRS-1 includes the Ethernet header (14 bytes), so the real maximum IP MTU is 9202 bytes. The CESNET2 network has been adapted to smoothly deploy CRS-1: we adjusted the IP MTU of 9202 bytes on all backbone router ports to unify it across the entire backbone.
- Poor SNMP performance with the CESNET2 SNMP measurement G3 systems. SNMP polling to obtain the data from the routers has high require-
Figure 5. CRS-1/16 Chassis: Power Load Zones and LC/SDR/Slot allocation

ments on CPU performance (OSR7609 are O.K.). Poor SNMP performance of CRS-1 was caused by an internal IPC procedure for environmental sensors (our chassis configuration has about 421 temperature, voltage, current and fan speed sensors). It was considered as a bug, already fixed.

The CRS-1/16 installation and migration process successfully passed all the NFRU tests that verified compatibility with the IP/MPLS and DWDM ONS15454
network layers, routing system management and monitoring, and hardware and network resiliency features. The NFRU tests were performed on the routing system connected to the live network under normal traffic load.

The resiliency tests covered the following topics:

- hardware availability – power supply failure
- hardware availability – OIR of Field-Replaceable Units (MSC and PLIM)
- hardware availability – fabric card failure
- hardware availability – RP fail-over (with/without NSF)
- network availability – core link failure
- software availability – process restart/crash (mpls ldp and ospf)

The status of CRS-1/16 and the entire CESNET2 network was monitored and checked. Packet losses were tested by iperf SW on the two Linux PCs connected by 10 GE to the both CRS-1/16 logical routers (PE-node and P-node). The test results demonstrated and confirmed the high availability features of the router.

3 Praha PoP Splitting

From the perspective of the CESNET2 network topology, the network core is resilient and fully redundant (see Figure 6). There are two PE Internet peering routers (R118 and R115) where the main and backup connections to the upstream ISP, NIX.CZ, and GN2 network are connected. Peering routers are double connected to core P routers (R119 and R105), where the circuits from the CESNET2 PoPs terminate. Various optimization techniques are implemented for network protocols such as MPLS LDP, OSPF and BGP to accomplish very quick convergence.

The main core routers are now located in the Praha PoP in the same computer room. The node is not resilient against rare outages, like power problems, cooling unit problems, etc. The same risk we experience with the key network servers (DNS, mail servers) which are resilient but again in the same computer room. The current Praha PoP encounters many limitations with the available space, power and cooling capacity.

Last year we started preparation for finding the availability of housing in the Praha area to split this main CESNET2 PoP into two geographically separated ones. The repeated tender in 2008 was successful, so we started the PoP splitting planning and preparation. We expect to move the PE Internet peering router R115 and the P router of the network core R105 to a new housing location (Praha II. PoP). Key network servers and some backbone optical circuits termination from the current PoP (Praha I.) will also be moved there. The PoP migration must be performed smoothly and is divided into the following phases:
Figure 6. Current CESNET2 IP/MPLS topology

1. PE Internet peering router and P router installation in Praha II.
   — Other routers will be used in parallel with the current R115 and R105. The new PE R114 (OSR7609-S) will be used with the more efficient RSP-720 processor. The P router R107 will be moved from the Praha I. PoP (is now available after the migration to CRS-1/16).
   — Planned to finish by mid-December 2008.

2. DWDM circuit Praha I. - Praha II.
   — DWDM technology will be used on the optical circuit between the PoPs. The main goal is to operate several data circuits on one optical fibre for various PoP interconnects (see Figure 7) to maintain logical network resiliency and to separate core MPLS and temporary 802.1Q data circuits. 32 channels DWDM Mux/Demux and DWDM pluggable optics in the routers will be used.
   — Planned to finish by mid-December 2008.

3. ONS 15454 ROADM node installation in Praha II. and its integration into main core DWDM ring
Most data circuits between the CESNET2 PoPs are provided by the main DWDM optical transmission system based on a ONS15454 platform. For the flexible move of these circuits from Praha I. to Praha II. there is a need to install a ROADM node in the Praha II. PoP and to integrate it with the main DWDM ring.

Optical circuit Praha I. - Praha II. For DWDM integration it will be geographically different from the first one.

ONS15454 ROADM network simulation has been performed. For correct DWDM operation, new DCU-100 units and a redesign of the link Praha-Hradec Králové will be needed.

Planned to finish in February 2009.

### 4. Selected optical circuit and services migration from Praha I. to Praha II.

- The circuits and services are provided by third party (Internet connectivity, optical lines lease).
- It must be provided step by step with the provider, CESNET and housing centre cooperation.
- Planned to finish in April 2009.

### 5. Smooth server migration

- The key network segments will be extended to the Praha II. PoP by dedicated data circuits and 802.1Q.
- Beginning planned for January 2009.

### 4 CESNET2 Network Resilience and Availability

CESNET2 network services availability and resiliency is the key motivation for backbone network deployment. In order to accomplish high availability CESNET2 network as a whole, we have to consider all the network layers and technologies. It begins with the dark fibre topology (especially the geographically diverse last miles to the PoPs), continues with the DWDM optical transmission technology and network, and ends with the IP/MPLS network. Redundancy and high availability are important features of the active network elements (DWDM nodes, routers, switches and others) and device service coverage with short disturbance clearance time.

Deployment of a new CL DWDM system on the optical lines Plzeň – Cheb – Most – Ústí nad Labem – Děčín, České Budějovice – Jindřichův Hradec – Jihlava – Brno and Letohrad – Opava (see Figure 1) allowed for backup circuits for these CESNET2 PoPs. All the PoPs above are now dual-home connected with the new PE routers C7606-S in the Jihlava and Most PoPs. There are also express 10 Gbps
backup optical circuits České Budějovice – Brno and Plzeň – Ústí n. Labem included in the main network core.

The replacement of the main PE Internet peering and P core routers by the new high availability and high performance routing system CRS-1/16 recently increased the availability of network services and overall network stability. There are many advanced features implemented in the modular IOS-XR which guarantee high availability:

- ISSU (In Service SW Upgrade) – non-service affected IOS-XR or particular processes upgrades
- NSF (Non-Stop Forwarding) – allows the forwarding of data packets to continue along known routes while the routing protocol information is being restored following RP fail-over. With NSF, peer networking devices do not experience routing flaps. During fail-over, the data traffic is forwarded through line cards while the standby RP assumes control from the failed one.
- Process restart/crash – automatic restart of crashed processes, with no impact on system operation
- Security protocol LPTS (Local Packet Transport Protocol) manages pack-
ets delivery to distributed processes across the system HW (LC CPU, RPs, DRPs). It secures the distributed control plane against DDoS attacks.

The rapid convergence of the running protocols used in the backbone network is a key for network availability and resilience guarantee. In the CESNET2 IP/MPLS network there are MPLS LDP, OSPF and BGP protocols, which have to rapidly converge in the case of line or router failure, to reroute the forwarded traffic.

In 2008 we tested and implemented rapid/sub-second convergence of the MPLS core backbone network CESNET2 (typically below one second) for the eventual failure of the line/router through rapid detection of disorders by using the BFD (Bidirectional Forwarding Detection) protocol.

The BFD is a detection protocol that is designed to provide fast forwarding path failure detection times for all media types, encapsulations, topologies, and routing protocols. In addition to fast forwarding path failure detection, BFD provides a consistent failure detection method. Using BFD to detect forwarding path failures at a uniform rate, rather than the variable rates for different routing protocol hello mechanisms, network profiling and planning is easier, and reconvergence time is consistent and predictable.

We configured BFD with the parameters of 3×0.1 sec using OSPF configuration / LDP Session Protection and optimization of the internal parameters of OSPF time increment, with the use of synchronization protocols OSPF, and LDP to avoid the transition phenomenon route to the “black hole” (blackholing avoidance) during the period of convergence of the two protocols.

It also increased the availability of the network (and its resilience to failures) with the implementation of NSF (Non-Stop Forwarding) with the SSO (Stateful switchover) for the Cisco OSR 7609 routers with redundant supervisors for the LDP, OSPF and BGP protocols. Using NSF with SSO we get the ability to route packets when the primary RP crashes and a router is being switched to the secondary one (RP fail-over) while keeping routing state information (graceful restart).

This feature minimizes (or completely eliminates) the forwarded packet loss (routing is typically interrupted for a period of about 0 - 3 sec on the affected Cisco 7600 router) to strongly enhance the stability of the network, since the original routing state information is replaced after the successful resynchronization with the NSF neighbours.
5 Future Plans

In the near term we plan to finish the Praha II PoP migration to improve the CESNET2 network availability even further. This includes a new ONS15454 ROADM node installation and integration into the main core DWDM ring and moving selected data circuits to this new PoP. The CESNET2 network topology must be adapted to the new PoP to ensure better network services availability.

The CRS-1/16 routing platform implementation in the Brno PoP will be our main goal in the near future. This is a fundamental prerequisite for the future testing and implementation of a 40 Gbps DWDM link between the Praha and Brno PoPs. Finally, the new CRS-1/16 will replace the currently overloaded peer-routing router R98 (OSR7609). We also plan to start the Brno PoP splitting process (both at the IP/MPLS and ONS15454 DWDM network layers).

The CL DWDM system is going to be delivered on the fibre optic lines Pardubice – Hradec Králové (as the extension of the current line Praha – Pardubice) and Brno – Zlín – Olomouc. The planned optical channels capacity is 10 Gbps and the IP/MPLS Zlín PoP will migrate to 10 Gbps. We further plan to upgrade the single-fibre lines Ostrava – Opava and Ostrava – Karviná to CL DWDM technology (1 Gbps channels only).

References


Abstract
This paper describes an experience with the deployment of the DWDM system, based on the open photonic devices family called CzechLight, within the CESNET2 network optical infrastructure in 2008. Emphasis is given on single fibre bidirectional DWDM systems. The key motivation for CL DWDM based systems deployment is a cost effective solution for gigabit (1-10Gbps) connectivity provisioning for small PoPs with the backup connectivity based on pure optical circuits. A NIL approach, i.e. optical line design without the inline components (e.g. amplifiers and chromatic dispersion compensators), was implemented.

Keywords: photonic transmission system, DWDM, NIL, single fibre bidirectional transmission, open photonic system

1 Introduction
The CzechLight (CL) family devices and their application, especially CL Amplifiers (CLA), have been extensively described in the past, e.g. [1], [2] and [3]. The CLAs can be easily deployed for multi-gigabit transmission over single fibre bidirectional links, as reported in [4]. The key motivation for single fibre systems is the provision of a cost effective (both OPEX and CAPEX) DWDM (Dense Wavelength Division Multiplex) system. This approach has been presented before, e.g. [5], and also deployed in other National Research and Educational Networks (NREN) [6]. The DWDM system functions over a single fibre similarly to a fibre pair system. Signals are multiplexed and demultiplexed or amplified in the same way as in a fibre pair system, with the difference that output and input signals must be mixed together before entering and after leaving the transmission line. This can be performed by different means, e.g. by using optical circulators, wavelength interleavers or WDM couplers. An example setup of bidirectional DWDM transmission over 210km based on circulators reported in [7] is shown in Figure 1.

When NIL (Nothing-in-Line) setup over large distance must be deployed, the issue of high backscattered power arises. To overcome this issue, different channels for each transmission direction must be used which excludes deployment of circulators. From an economical point of view, the best option is to use simple WDM couplers.
The average price of a dark fibre pair lease is about 0.5 EUR/meter/year and the price of a single fibre lease is about 40 % lower, 0.3 EUR/meter/year [8]. The single fibre bidirectional transmission brings the necessity of more complex hardware, but the average increase in the price of transmission equipment is about 0.004 EUR/meter/year for NIL single fibre lines and about 0.002 EUR/meter/year for multi span lines. These average data are based on examples of lines deployed by open photonic transmission systems mentioned in [9]. Together with a slight increase in the price of hardware, the only drawback of a single fibre solution is the reduction of transmission channels (by one half, compared with the fibre pair system). Nevertheless, while we are relying on mature EDFA technology, 32 or 40 bidirectional channels with 100 GHz spacing can be easily created exploiting both C and L bands.

Considering that present research networks typically operate a small number of channels (below 10) and for “rural lines” there are no needs or plans to deploy over 32 channels this reduction should not be an obstacle.
2 “Regular” Fibre Pair Optical Lines

2.1 České Budějovice – Jihlava – Brno

In 2008 the CL DWDM deployment of a České Budějovice – Jihlava – Brno line was successfully completed with the following goals:

— full 10Gbps backup for České Budějovice PoP.
— 1Gbps backup connectivity for Jihlava PoP.
— 1Gbps backup connectivity for Jindřichův Hradec PoP.
— simple provisioning of new optical channels between any PoP

The line was designed in the form of two independent spans depicted in Figure 2. The Jihlava PoP is a full add/drop node allowing more freedom for future upgrades. It means that all wavelengths are dropped or added. Amplification is performed by CLA PB01F amplifiers and chromatic dispersion (CD) is compensated by un-channelized fibre Bragg gratings (FBGs). They support easy future upgrade, allowing deployment of spectrally broader signals. These signals could be induced by possible fast transmissions speeds, e.g. 40 or 100 Gbps. Signals are multiplexed and demultiplexed by 32 channel athermal arrayed waveguide gratings (AAWGs) which open up channel planning possibilities.

Currently Jihlava is a 1Gbps PoP, with a future upgrade to 10G possible anytime, just by an exchange of transceivers in the client devices, i.e. IP/MPLS routers. Connectivity for the Jindřichův Hradec PoP is provided by an OADM node at Jindřichův Hradec railway station, where 3 lambdas are dropped and transported to/from the Jindřichův Hradec PoP.

2.2 Brno – Ostrava

The upgrade of the Brno – Ostrava line has brought a full 10Gbps backup connection for the Ostrava PoP and a reduction in latency of the line moderately loaded by almost 2.5 Gbps of traffic.

The line deployment is shown in Figure 3. To overcome a huge attenuation of the 235km long line, the CLA PB02F amplifiers (with saturated output power about 27 dBm) were deployed. CD is compensated for by un-channelized FBGs for future upgradeability. Signals are multiplexed and demultiplexed by 32 channel AAWGs for achieving freedom in channel planning.
2.3 Praha – Pardubice

In 2008 a leased fibre pair line was upgraded from a “grey” single 1 Gbps to 10Gbps DWDM system using the “lit fibre approach”, see [8]. It means that deployment was in the hands of the fibre pair owner, but all capacity remains with the customer (i.e. CESNET). The line deployment is shown in Figure 4. To overcome attenuation of this nearly 190 km long line, the CLA PB02F amplifiers were deployed. CD is compensated by un-channelized FBGs for future upgradeability. Signals are multiplexed and demultiplexed by 32 channel AAWGs for freedom in channel planning.

The Pardubice PoP is planned to be a full ADD/DROP node and upgrade of the span to Hradec Králové is in progress.

3 Single Fibre Bidirectional Optical Lines

3.1 Plzeň-Cheb-Most-Ústí n. Labem

The network of single fibre lines connecting small PoPs in West and North-West Bohemia (Cheb, Most, Děčín) has been extended and the DWDM system has been designed to provide backup connectivity for these PoP and also all-optical 10Gbps capable paths between the Plzeň and Ústí nad Labem PoPs.

The planned deployment is shown in Figure 5. An amplification is provided by CLA PB01 F amplifiers as a relatively low channel number is considered. Over the Plzeň – Cheb – Most – Ústí n. Labem lines the CD is compensated by un-channelized FBGs for future upgradeability. As there is no demand for 10Gbps connections in Děčín, the CD of the Most – Děčín – Ústí n. Labem line remains uncompensated. 8-channel MUX/DEMUXes based on thin filter technology were used. The Děčín PoP was designed as a passive OADM node similar to the planned node in Mariánské Lázně. The Most PoP was designed as a full ADD/DROP node to allow crossing of three directions.

The “lit fibre” approach was used and upgrades were requested from different fibre owners. In November 2008 the Plzeň – Cheb and Ústí nad Labem – Most lines were upgraded and came into operation. For example, the upgrade of the Plzeň – Cheb line has allowed high speed interconnection of backup servers. Traffic invoked by large backup is shown in Figure 6. Upgrades of the remaining lines are in progress.
3.2 Letohrad-Opava

The reason for deployment of the Letohrad – Opava line was to provide the a second connection for the Opava PoP.

The line setup is shown in Figure 7. An amplification is provided by CLA PB01 F amplifiers as a relatively low number of channels is considered. Only 1Gbps speed is considered, therefore the CD remains uncompensated. 6-channel MUX/DEMUXes based on thin filter technology were used.

As the Letohrad PoP is an optical transport (ROADM) node only (it has no IP equipment), lambdas are transported as alien wavelengths over the main DWDM Mux/Demux 32 channels.
Figure 3. Brno – Ostrava line

Figure 4. Praha – Pardubice line with planned extension to Hradec Králové

system based on Cisco ONS15454 MSTP technology from/to Hradec Králové. Unfortunately, outgoing signal levels from the ONS 15454 MSTP system are too weak (about -12 dBm) which requires an additional amplification and the upgrade of the Letohrad node has been ordered. The line currently operates in an experimental regime with limited monitoring possibilities.

4 Future Plans

The CESNET2 backbone network optical topology is depicted in Figure 8. The main DWDM backbone system is based on commercially available technology (Cisco ONS 15454 MSTP with the ROADM) in the optical network core. The DWDM system interconnects the most important PoPs at the optical level, when there is a need for more optical transport channels (between the PoPs or for research projects). It also allows the flexible optical channels provisioning using the ROADM “on demand”. The CL DWDM systems on some backbone lines
effectively complements the main DWDM system and are effective (in terms of CAPEX, OPEX including power consumption) in comparison with commercially available DWDM systems. Both ONS15454 MSTP and CL DWDM systems support the “alien” wavelength transport via the entire optical backbone network.

We plan to implement CL DWDM systems on new optical lines during future CESNET2 deployment stages. In 2009 we plan to deliver a 10 Gbps connectivity and backup connectivity to the Zlín PoP, and an upgrade of Brno – Zlín – Olomouc fibre pair lines has been designed. The “lit fibre service” approach will be used again.

Zlín was designed to be a full ADD/DROP node where signals will be multiplexed and demultiplexed by 32-channel AAWGs for freedom in future channels planning. CD will be compensated by un-channelized FBGs for future upgradeability. Attenuation of the longer Brno – Zlín span is compensated by CLAPB02F amplifiers. The Zlín – Olomouc span will be operated by CLAPB01F devices.

For a 10 Gbps backup connectivity of the Pardubice PoP, the upgrade of the

---

**Figure 5.** Single fibre network in west Bohemia
fibre pair Pardubice – Hradec Králové has also been designed.

The Pardubice PoP is considered to be a full ADD/DROP node and signals will be MUXed/DEMUXed by 32-channel AAWGs for freedom in future channels planning. As the line length is only about 30km, the CD will not be compensated.
Figure 8. CESNET2 network optical topology

Figure 9. Brno – Zlín – Olomouc
and one-side amplification will be provided by a single CLADI02F device.

Our plan for 2009 is to implement and test the new CL DWDM technology components in the CESNET2 operational network. The first CL DWDM ROADM and CLS (CL optical switches) will be deployed on the CL DWDM optical lines.

References


¹ http://www.ces.net/events/20050516/pr/karasek-radil-vojtech.ppt
² http://www.ces.net/events/20060529/pr/vojtech.ppt
³ http://www.ces.net/events/20060529/pr/kugler.pdf
⁴ http://www.ces.net/events/cef2007/p/altmannova.ppt
QoS Design and Implementation in the CESNET2 MPLS-based Backbone

Pavel Smrha, Josef Verich

Abstract

This paper deals with the design and implementation of QoS in the CESNET2 MPLS backbone composed of Cisco 7600 and Cisco CRS-I routers. The technical approach is to implement DiffServ within the MPLS-based CESNET2 backbone using the E-LSP point-to-cloud (destination unaware) QoS model for six classes of traffic aggregates with different QoS characteristics. To ensure DSCP transparency for transported IPv4/IPv6 user traffic, the so-called "short pipe" MPLS DiffServ tunnelling mode is deployed. The QoS implementation for IPv4 and IPv6 traffic currently uses six classes of service: Real-Time (EF-PHB), Network Control (AF-PHB), Video (AF-PHB), Critical Traffic (AF-PHB), Best Effort (default PHB), and Less than Best Effort (AF-PHB).

Keywords: Quality of Service, DiffServ, MPLS

1 Introduction

The basic goal of Quality of Service (QoS) is to provide service differentiation among IP packets in the network. Such a service differentiation is noticeable during periods of network congestion (i.e. in the case of contention for resources) and results in different levels of network performance.

The initial phase of QoS implementation in the CESNET2 network was completed in 2005 [15]. This document describes the second network QoS deployment phase in the CESNET2 MPLS backbone, the Czech National Research and Educational Network (NREN). It covers mainly service class restructuring, IPv6 support and unification of QoS configuration for new hardware platforms or modules (e.g. CRS-1 and Cisco 7600 ES20 modules). Such an implementation of preferential service for a class of traffic not only requires technical support to achieve appropriate network behaviour, but also raises other issues related to establishing an acceptable overall QoS policy framework [8].

The main goals of this network QoS stage of implementation in the CESNET2 MPLS backbone are the following:

— To define a simple yet consistent QoS policy framework for the CESNET2 MPLS backbone, which is also easy to implement, provision, and manage.
To provide the necessary QoS support for CESNET2 IPv4/IPv6 transit traffic to enable end-to-end QoS-based connectivity for users within different network domains (e.g. university MANs, regional networks etc.).

To ensure DSCP transparency for all SLS-conformant IPv4/IPv6 user traffic while transiting the CESNET2 MPLS backbone (i.e. the original value of the DSCP field within the customer in-profile IPv4/IPv6 packets is not modified as they are transported through the CESNET2 MPLS backbone).

To ensure compliance with the GÉANT2 Premium IP class of service.

To be compatible with other GÉANT2 interdomain classes of service (e.g. Best Effort and Less than Best Effort), while providing new additional useful classes of service not (yet) implemented in GÉANT2.

To design a QoS configuration as unified as possible across all routers and other network devices in the CESNET2 MPLS backbone (i.e. Cisco 7600 OSR/Sup720/PFC3BXL/CXL and Cisco CRS-1).

The technical approach is to implement DiffServ in the CESNET2 MPLS-based backbone using the E-LSP point-to-cloud destination unaware QoS model for a few classes of traffic aggregates with different QoS characteristics. To ensure DSCP transparency for transported IPv4/IPv6 user traffic the so-called “Short-Pipe” MPLS DiffServ tunnelling mode has been deployed\(^1\).

In every case (with/without congestion) there should be guarantees for all traffic classes for their (provisioned) bandwidth as well as other qualitative characteristics of their class (e.g. delay, jitter, packet loss). In case of no congestion it is required that all classes\(^2\) can use the otherwise unused available bandwidth in proportion to their weight (even if they utilize more bandwidth than guaranteed for their class by provisioning).

## 2 Quality of Service on GÉANT2

This chapter summarizes current implementation of QoS on GÉANT2 using information available from the DANTE/GÉANT2 web site [13].

Currently, three service classes are offered that can be used for traffic transiting the GÉANT2 network:

\(^{1}\) Due to the possibility of penultimate hop popping (PHP) when MPLS/BGP VPN technology is not used and only one MPLS label is present.

\(^{2}\) With the possible exception of specific hardware or configuration limitations (e.g. various Cisco IOS “priority” classes implementations) or other requirements (e.g. for security purposes).
— The Premium IP service provides guarantees on one-way delay, jitter and packet loss. Applications with real time constraints may benefit from this service; for instance video conferencing.
— The Best Effort service does not provide any performance guarantees. However, the goal is to give a fair share to each traffic flow. Best effort is the default class of service and is useful for traffic from non-real-time applications such as FTP.
— The Less than Best Effort (scavenger) service utilizes the remaining capacity that is not used by best effort and premium IP traffic. This service may be useful to transfer vast amounts of (non real-time) data for applications such as GRID computing without adversely affecting other traffic.

Both Best Effort (BE) and Less than Best Effort (LBE) services are available to peer networks. The Premium IP (PIP) service, however, is provided only upon reservation.

Table 1. DSCP to service class mapping on GÉANT2.

<table>
<thead>
<tr>
<th>Service Class Name</th>
<th>DSCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Premium IP (PIP)</td>
<td>EF</td>
</tr>
<tr>
<td>Best Effort (BE)</td>
<td>0</td>
</tr>
<tr>
<td>Less than Best Effort (LBE)</td>
<td>CS8</td>
</tr>
</tbody>
</table>

3 Quality of Service in CESNET2

3.1 DiffServ Marking Schemes

Currently there are at least two consistent DiffServ marking schemes available for academic user traffic: the so called Cisco QoS Baseline [9], which was adopted by some universities in the past, and RFC 4594 Configuration Guidelines for DiffServ Service Classes [2].

3.1.1 Cisco QoS Baseline

While the IETF RFC standards provided a consistent set of per-hop behaviours for applications marked to specific DSCP values, they had never specified which application should be marked to which DSCP value until 2006. This confusion led Cisco to put forward a standards-based marking recommendation in their strategic architectural QoS Baseline document [9] in 2002. Eleven different application classes that could exist within the organization were examined and extensively
profiled, and then matched to their optimal RFC-defined Per-Hop Behaviours (PHBs). The QoS Baseline is a strategic document designed to unify QoS within Cisco. It provides uniform, standards-based recommendations to help ensure that QoS products, designs, and deployments are unified and consistent. The QoS Baseline defines up to 11 classes of traffic that may be viewed as critical to a given organization. A summary of these classes and their respective standards-based markings and recommended QoS configurations are shown in Table 2:

- **IP Routing** class is intended for IP routing protocols, such as Border Gateway Protocol (BGP), Open Shortest Path First (OSPF).
- **Voice** refers to VoIP bearer traffic only (and does not include call-signalling traffic).
- **Interactive-Video** refers to IP video-conferencing.
- **Streaming Video** is aimed at either unicast or multicast uni-directional video.
- **Mission-Critical Data** (locally defined) class is intended for a subset of Trans- actional Data applications that contribute most significantly to the business objectives (this is a non-technical assessment).
- **Call-Signalling** class is intended for voice and/or video signalling traffic, such as Skinny, SIP, H.323.
- **Transactional Data** class is intended for foreground, user-interactive applications such as database access, transaction services, interactive messaging, and preferred data services.
- **Network Management** class is intended for network management protocols, such as SNMP, Syslog, DNS.
- **Bulk Data** class is intended for background, non-interactive traffic flows, such as large file transfers, content distribution, database synchronization, email and backup operations.
- **Best Effort** class is the default class. Unless an application has been assigned a preferential/deferential service, it remains in this class. Most organizations have hundreds of applications on their networks; the majority of which will fall into the Best Effort service class.
- **Scavenger** class is based on RFC 3662 and an Internet 2 draft that defines a “Less-than-Best Effort” service. In the event of link congestion, this class will be dropped the most aggressively.

Standard-based marking recommendations allow for better integration with service-provider offerings as well as other internetworking scenarios. In Cisco IOS Software, rate-based queueing translates to CBWFQ; priority queueing is LLQ. DSCP-Based WRED (based on RFC 2597) drops AFx3 before AFx2, and in turn drops AFx2 before AFx1. RSVP is recommended (whenever supported) for Voice and/or Interactive-Video admission control in Cisco products that support QoS.
Table 2. Summary of QoS Mechanisms Used for Each Cisco Baseline Service Class.

<table>
<thead>
<tr>
<th>Service class</th>
<th>DSCP</th>
<th>Conditioning</th>
<th>PHB</th>
<th>Queueing</th>
<th>AQM</th>
</tr>
</thead>
<tbody>
<tr>
<td>routing</td>
<td>CS6</td>
<td>sr+bs</td>
<td>RFC 2474 rate</td>
<td>RED</td>
<td></td>
</tr>
<tr>
<td>voice</td>
<td>EF</td>
<td>sr+bs</td>
<td>RFC 3246 priority</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>interactive video</td>
<td>AF4x</td>
<td>trTCM</td>
<td>RFC 2597 rate</td>
<td>WRED/DSCP</td>
<td></td>
</tr>
<tr>
<td>streaming video</td>
<td>CS4</td>
<td>sr+bs</td>
<td>RFC 2474 rate</td>
<td>RED</td>
<td></td>
</tr>
<tr>
<td>mission-critical data</td>
<td>AF3x</td>
<td>srTCM</td>
<td>RFC 2597 rate</td>
<td>WRED/DSCP</td>
<td></td>
</tr>
<tr>
<td>call signalling</td>
<td>CS3</td>
<td>sr+bs</td>
<td>RFC 2474 rate</td>
<td>RED</td>
<td></td>
</tr>
<tr>
<td>transactional data</td>
<td>AF2x</td>
<td>srTCM</td>
<td>RFC 2597 rate</td>
<td>WRED/DSCP</td>
<td></td>
</tr>
<tr>
<td>network management</td>
<td>CS2</td>
<td>sr+bs</td>
<td>RFC 2474 rate</td>
<td>RED</td>
<td></td>
</tr>
<tr>
<td>bulk data</td>
<td>AF1x</td>
<td>trTCM</td>
<td>RFC 2597 rate</td>
<td>WRED/DSCP</td>
<td></td>
</tr>
<tr>
<td>best effort</td>
<td>CS0</td>
<td>—</td>
<td>RFC 2474 rate</td>
<td>RED</td>
<td></td>
</tr>
<tr>
<td>scavenger</td>
<td>CS1</td>
<td>—</td>
<td>RFC 3662 rate</td>
<td>RED</td>
<td></td>
</tr>
</tbody>
</table>

features and will use these QoS Baseline recommendations for marking, scheduling, and admission control.

The QoS Baseline recommendations are intended as a standards-based guideline for users · not as a mandate. Users do not have to deploy all 11 traffic classes, but may start with simple QoS models and expand over time as business needs arise.

3.1.2 RFC 4594: Configuration Guidelines for DiffServ Service Classes

Informational RFC 4594 describes service classes configured with DiffServ (DS) and recommends how to use and construct them using Differentiated Services Code Points (DSCPs), traffic conditioners, Per-Hop Behaviours (PHBs), and Active Queue Management (AQM) mechanisms. There is no intrinsic requirement that particular DSCPs, traffic conditioners, PHBs, and AQM be used for a certain service class, but as a policy and for interoperability it is useful to apply them consistently.

Table 3 defines the recommended relationship between service classes and DSCP assignment with application examples. It is recommended that this relationship be preserved end-to-end.

Table 4 provides a summary of DiffServ QoS mechanisms that should be used for the defined service classes. According to what applications/services need to be differentiated, network administrators can choose the service class(es) that need to be supported in their network.
Table 3. DSCP to service class mapping [2].

<table>
<thead>
<tr>
<th>Service class</th>
<th>DSCP</th>
<th>Application example</th>
</tr>
</thead>
<tbody>
<tr>
<td>network control</td>
<td>CS6</td>
<td>network routing: BGP, OSPF, ISIS, RIP, CR-LDP, RSVP-TE</td>
</tr>
<tr>
<td>telephony</td>
<td>EF</td>
<td>IP telephony bearer: VoIP, CEoIP, virtual wire</td>
</tr>
<tr>
<td>signalling</td>
<td>CS5</td>
<td>IP telephony signalling: SIP H.323, H.248, MGCP, SIP-T</td>
</tr>
<tr>
<td>multimedia conferencing</td>
<td>AF4x</td>
<td>H.323/V2 video conferencing (adaptive): H.323/v2</td>
</tr>
<tr>
<td>real-time interactive</td>
<td>CS4</td>
<td>video conferencing and interactive gaming</td>
</tr>
<tr>
<td>multimedia streaming</td>
<td>AF3x</td>
<td>streaming video and audio on demand</td>
</tr>
<tr>
<td>broadcast video</td>
<td>CS3</td>
<td>broadcast TV &amp; live events</td>
</tr>
<tr>
<td>low-latency data</td>
<td>AF2x</td>
<td>client/server transactions, Web-based and ERP: SNA/DLSw, SAP</td>
</tr>
<tr>
<td>OAM</td>
<td>CS2</td>
<td>OAM&amp;P: SNMP, TFTP, FTP, Telnet, SSH</td>
</tr>
<tr>
<td>high-throughput data</td>
<td>AF1x</td>
<td>store and forward applications: FTP, email</td>
</tr>
<tr>
<td>Standard</td>
<td>CS0</td>
<td>Undifferentiated applications: DNS, DHCP/Bootp, NTP</td>
</tr>
<tr>
<td>low-priority data</td>
<td>CS1</td>
<td>any flow that has no bandwidth assurance (Scavenger)</td>
</tr>
</tbody>
</table>

3.1.3 Recommended DSCP Marking Scheme

When comparing both recommendations, Cisco Baseline and RFC 4594, we can find certain incompatibilities, both in the number and type of classes proposed and in their marking, as Table 5 shows.

The major changes include; one application class was removed, two marking changes were applied, and two new application classes were added [7]:

- The QoS Baseline locally-defined Mission-Critical Data class has been deleted from RFC 4594.
- The QoS Baseline marking recommendation of CS4 for Streaming Video has been changed in RFC 4594 to mark Multimedia Streaming to AF31.
- The QoS Baseline marking recommendation of CS3 for Call Signalling has been changed in RFC 4594 to mark Call Signalling to CS5.
Table 4. Summary of QoS Mechanisms Used for Each Service Class [2].

<table>
<thead>
<tr>
<th>Service class</th>
<th>DSCP</th>
<th>Condit.</th>
<th>PHB</th>
<th>Queueing</th>
<th>AQM</th>
</tr>
</thead>
<tbody>
<tr>
<td>network control</td>
<td>CS6</td>
<td>sr+bs</td>
<td>RFC 2474</td>
<td>rate</td>
<td>yes</td>
</tr>
<tr>
<td>telephony</td>
<td>EF</td>
<td>sr+bs</td>
<td>RFC 3246</td>
<td>priority</td>
<td>no</td>
</tr>
<tr>
<td>signalling</td>
<td>CS5</td>
<td>sr+bs</td>
<td>RFC 2474</td>
<td>rate</td>
<td>no</td>
</tr>
<tr>
<td>multimedia conferencing</td>
<td>AF4x</td>
<td>trTCM</td>
<td>RFC 2597</td>
<td>rate</td>
<td>yes, per DSCP</td>
</tr>
<tr>
<td>real-time interactive</td>
<td>CS4</td>
<td>sr+bs</td>
<td>RFC 2474</td>
<td>rate</td>
<td>no</td>
</tr>
<tr>
<td>multimedia streaming</td>
<td>AF3x</td>
<td>trTCM</td>
<td>RFC 2597</td>
<td>rate</td>
<td>yes, per DSCP</td>
</tr>
<tr>
<td>broadcast video</td>
<td>CS3</td>
<td>sr+bs</td>
<td>RFC 2474</td>
<td>rate</td>
<td>no</td>
</tr>
<tr>
<td>low-latency data</td>
<td>AF2x</td>
<td>srTCM</td>
<td>RFC 2597</td>
<td>rate</td>
<td>yes, per DSCP</td>
</tr>
<tr>
<td>OAM</td>
<td>CS2</td>
<td>sr+bs</td>
<td>RFC 2474</td>
<td>rate</td>
<td>yes</td>
</tr>
<tr>
<td>high-throughput data</td>
<td>AF1x</td>
<td>trTCM</td>
<td>RFC 2597</td>
<td>Rate</td>
<td>Yes per DSCP</td>
</tr>
<tr>
<td>standard</td>
<td>CS0</td>
<td>—</td>
<td>RFC 2474</td>
<td>rate</td>
<td>yes</td>
</tr>
<tr>
<td>low-priority data</td>
<td>CS1</td>
<td>—</td>
<td>RFC 3662</td>
<td>rate</td>
<td>yes</td>
</tr>
</tbody>
</table>

— A new video class has been added to RFC 4594: Real-Time Interactive, which is to be marked CS4. This was done to differentiate between lower-grade desktop telephony (referred to as Multimedia Conferencing) and higher-grade videoconferencing and telepresence.

— Another new video class has been added to RFC 4594: Broadcast Video, which is to be marked CS3. This was done to differentiate between lower-grade desktop video streaming (referred to as Multimedia Streaming) and higher-grade Broadcast Video applications. Multimedia Streaming uses the AF3 class and is subject to markdown policies, while Broadcast Video uses the CS3 class and is not subject to markdown.

Some incompatibilities of marking can be partially eliminated under certain circumstances through the support of dual marking, e.g. for the aggregated signalling class, it is possible (under an absence of support of conflict Broadcast Video class marked also as CS3) to support simultaneously CS3 and CS5 within the transit DS domain according to the Cisco Baseline model because CS5 is not used for any basic class in that model. Unfortunately, it is then necessary to perform re-classification/remarking to the selected semantics at the border of DS domain for conflict marking in general (CS3, CS4 and AF3x). As the informational RFC 4594 recommendation offers finer granularity of support for multimedia applications and is rather oriented to general providers of a wide scale of services, we have chosen the variant of Cisco Baseline model for the proposed QoS classes of DS domain of the CESNET2 network environment. On one hand it seems to correspond better to the requirements for QoS in current academic user traffic thanks to the concept of the usually required Mission-Critical Data service class. On the
Table 5. RFC 4594 vs. Cisco Baseline Service Class Comparison.

<table>
<thead>
<tr>
<th>RFC 4954 service class</th>
<th>DSCP</th>
<th>Cisco baseline service class</th>
<th>DSCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Control</td>
<td>CS6</td>
<td>Routing</td>
<td>CS6</td>
</tr>
<tr>
<td>Telephony</td>
<td>EF</td>
<td>Voice</td>
<td>EF</td>
</tr>
<tr>
<td>Signalling</td>
<td>CS5</td>
<td>Call Signalling</td>
<td>CS3</td>
</tr>
<tr>
<td>Multimedia Conferencing</td>
<td>AF4x</td>
<td>Interactive Video</td>
<td>AF4x</td>
</tr>
<tr>
<td>Real-Time Interactive</td>
<td>CS4</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>--</td>
<td>—</td>
<td>Mission-Critical Data</td>
<td>AF3x</td>
</tr>
<tr>
<td>Multimedia Streaming</td>
<td>AF3x</td>
<td>Streaming Video</td>
<td>CS4</td>
</tr>
<tr>
<td>Broadcast Video</td>
<td>CS3</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Low-Latency Data</td>
<td>AF2x</td>
<td>Transactional Data</td>
<td>AF2x</td>
</tr>
<tr>
<td>OAM</td>
<td>CS2</td>
<td>Network Management</td>
<td>CS2</td>
</tr>
<tr>
<td>High-Throughput Data</td>
<td>AF1x</td>
<td>Bulk Data</td>
<td>AF1x</td>
</tr>
<tr>
<td>Standard</td>
<td>CS0</td>
<td>Best Effort</td>
<td>CS0</td>
</tr>
<tr>
<td>Low-Priority Data</td>
<td>CS1</td>
<td>Scavenger</td>
<td>CS1</td>
</tr>
</tbody>
</table>

On the other hand it supports more consistent implementation of QoS in the environment of IP and MPLS networks implemented on Cisco Systems devices (which can be very important particularly because of some limitations in implementation of MPLS QoS in different hardware platforms of this vendor).

3.2 CESNET2 Service Classes

We propose six service classes for traffic transiting the CESNET2 MPLS-based backbone:

- **Real-Time (RT)** service provides guarantees on bandwidth, one-way delay, jitter and packet loss. This is intended to be the same service as the one referred to in the context of the GÉANT2 Premium IP (PIP) service. Applications with real time constraints may benefit from this service; for instance, Voice over IP (VoIP), real-time data from remote sensors.

- **Network Control (NC)** service provides guaranteed bandwidth for necessary network control traffic (e.g. routing updates) with relatively low delay and low packet loss. This service is intended to be considered rather a CESNET2 intradomain service, but in special cases it can also be used as an interdomain working to ensure stability when required (e.g. to be more resilient to congestion when using dynamic routing protocols with neighbouring DS domains of interconnected universities).

- **Video** service is intended as a preferential service for inelastic applications requiring guaranteed bandwidth, (much) better delay performance and very
low packet loss than the ones offered by BE but for which the tight constraints on delay and jitter offered by RT are not necessary. Some kind of videoconferencing, streaming delivery of non-interactive audio-video material are candidate applications. Traffic of this type is expected to form a relatively large proportion of total traffic.

- **Critical Traffic** (CT) service is intended as a preferential service for elastic (data) applications requiring guaranteed bandwidth with better delay performance and lower packet loss than the ones offered by BE but for which the tight constraints on delay and jitter offered by RT are not necessary. Interactive and transactional data applications and various signalling protocols are candidate applications. Traffic of this type is expected to form a relatively large proportion of total traffic.

- **Best Effort** (BE) service does not provide any performance guarantees. However, the goal is to give a fair share to each traffic flow. This is intended to be the same service as the one referred to in the context of the GÉANT2 BE service. Best effort is the default class of service and is useful for traffic from non-real-time applications such as FTP or email.

- **Less than Best Effort** (LBE) service will utilize the capacity that is not used by other traffic classes. This is intended to be the same service as the one referred to in the context of the GÉANT2 LBE service or as the *Scavenger Service* (QBSS) in QBone/Internet2/Abilene. This service may be useful to transfer vast amounts of (non real-time) data for applications such as GRID computing or backups without adversely affecting other traffic. This service can also be used to carry some excess traffic based on the results of various policing instead of dropping it, just to give it a “last resort” chance to be transferred through the network if (and only if) the network resources are available without affecting other higher service classes.

### 3.3 CESNET2 MPLS-backbone QoS Policy Framework

#### 3.3.1 Network Model

The policy framework defines Exp-LSP DiffServ technology with point-to-cloud destination-unaware model as a basis for QoS services to be deployed as part of the second phase of the CESNET2 QoS project. To ensure DSCP transparency the so called “Short-Pipe” MPLS DiffServ tunnelling mode will be deployed: the ingress PE router at the CESNET2 MPLS DiffServ domain border derives the EXP bits from the original DSCP of the incoming trusted IP packet, the egress PE router uses this original DSCP of the underlying IP packet (which remains unchanged while transiting the CESNET2 MPLS DiffServ domain) instead of
the (possibly changed) EXP marking used by intermediate P routers. Due to the possibility of penultimate hop popping (PHP) in case of not using MPLS/BGP VPN technology (when only one MPLS label is present) even the second-to-last router inside the CESNET2 MPLS DiffServ domain may use this original DSCP of the underlying IP packet for its forwarding decision.

The main feature of this model is that each router in the network processes packets in a differentiated manner: the differentiation being made on the basis of the value of the DSCP field of a packet at ingress/egress network points and the EXP field at intermediate network points, disregarding a destination address (path) of a packet. This feature results in a simplicity of deploying and managing QoS services but at the same time it has a negative side effect when a great amount of higher QoS classes’ flows go to the same output interface of the router, so that an interface becomes congested by these particular classes of traffic. The point-to-cloud model cannot prevent this situation completely, as routers and network administrators don’t check paths of flows and network users have the right to direct their flows wherever they like. To decrease the probability of this situation and instead increase the probability of providing a stable QoS service, the border routers of the CESNET2 MPLS DiffServ domain should police the higher level QoS classes to maintain them at relatively low levels conforming to agreed traffic contracts (SLA).

CESNET2 compliant DSCP marking must be provided by the neighbouring DiffServ domain at CESNET2 ingress to ensure easy QoS implementation, provisioning and management within the CESNET2 DiffServ domain. A preventive admission control method to keep DiffServ model working properly is to limit higher level QoS traffic within each aggregated class of service according to the SLA by policing the exceeding/violating portion of such traffic (and typically re-marking it to lower level QoS classes such as BE or LBE) at the ingress (or egress when required) of the CESNET2 MPLS DiffServ domain. As a result no tedious management of otherwise necessary access control lists of authorized sources for each particular QoS traffic class is required.

### 3.3.2 Provisioning

In considering the provisioning of the links constituting intra- and interdomain boundaries, several capacity limits can be distinguished [8]:

- **Absolute Link Capacity (ALC):** The transmission capacity of the physical link.
- **CESNET2 DiffServ Limit (C2DSL):** The maximum proportion of a given traffic class allowed on any link. If too much traffic of a given class is allowed onto a link, then the qualitative characteristics (e.g. jitter and delay) on the individual constituent flows will increase and the behaviour will tend to become
more like BE. In the extreme case with all the traffic allowed to be e.g. EF, the behaviour would revert to BE. The CESNET2 DiffServ traffic limit is the maximum proportion of a traffic class which should be admitted on a link, and it is determined by the need for this traffic class to operate correctly for the traffic mix for which this traffic class is intended. This limit is important because if more than the absolute amount represented by this proportion is needed on a given link, then a larger capacity link is needed in order to maintain the particular characteristics class of service.

— Operational Allocation (OA): The operating proportion determined as being available for the given traffic class on a given link. At any particular time, the provisioned traffic of this class, i.e. the amount admitted for the link,
may be less than the CESNET2 DiffServ limit for this traffic class, since the demand within the CESNET2 DiffServ domain may not require the maximum allocation, or it may be decided to limit it for other reasons.

— *Operational Policing Limit (OPL):* The operating policing parameters in force for the traffic class on a link. The purpose of policing traffic as it arrives is to ensure that the agreed provision of capacity for traffic using a particular service is not exceeded. Even in an environment where networks trust each other, such policing is generally desirable at least as the first level of defence against malicious traffic starving lower level service traffic.

**Dimensioning intradomain provisioning**

The following intradomain provisioning of the *CESNET2 DiffServ Limit* is required within the CESNET2 MPLS DiffServ domain:

— At most 33% of the *absolute link capacity* can be allocated for the Real-Time class of service.
— At least 25% of *absolute link capacity* must be left for the Best Effort class of service (i.e. remain unallocated to other service classes).

Table 6 shows the CESNET2 MPLS DiffServ intradomain operational allocation of reserved bandwidth and proposed PHB for supported QoS classes of services.

<table>
<thead>
<tr>
<th>Service class name</th>
<th>PHB</th>
<th>OA bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-Time</td>
<td>EF</td>
<td>25%</td>
</tr>
<tr>
<td>Network Control</td>
<td>AF</td>
<td>2%</td>
</tr>
<tr>
<td>Video</td>
<td>AF</td>
<td>20%</td>
</tr>
<tr>
<td>Critical Traffic</td>
<td>AF</td>
<td>25%</td>
</tr>
<tr>
<td>Best Effort</td>
<td>Default</td>
<td>25%</td>
</tr>
<tr>
<td>Less than Best Effort</td>
<td>AF</td>
<td>3%</td>
</tr>
</tbody>
</table>

**Dimensioning inter-domain provisioning**

Dimensioning inter-domain provisioning of the CESNET2 MPLS DiffServ domain with other QoS domains must be compliant with CESNET2 intradomain provisioning policy. It mainly means that the aggregated incoming and outgoing traffic of the particular QoS class must not negatively influence the required PHB of any network node within the CESNET2 MPLS DiffServ domain.

The QoS SLS should contain the following mandatory traffic parameters for each higher level class of service:
QoS Design and Implementation in the CESNET2 MPLS-based Backbone

Ingress Committed Rate within the class: ICR\textsubscript{class},
Ingress Burst Conform within the class: IBC\textsubscript{class},
Ingress Burst Exceed within the class: IBE\textsubscript{class},
and the following optional traffic parameters:

Egress Committed Rate within the class: ECR\textsubscript{class},
Egress Burst Conform within the class: EBC\textsubscript{class},
Egress Burst Exceed within the class: EBE\textsubscript{class},
and aggregated traffic (the whole traffic for all classes of service including higher level service classes and lower level service classes):

Ingress Committed Rate: ICR,
Ingress Burst Conform: IBC,
Ingress Burst Exceed: IBE,
optionally:
Egress Committed Rate: ECR,
Egress Burst Conform: EBC,
Egress Burst Exceed: EBE.

Such an SLS is capable of providing the necessary parameters for properly adjusting the srTCM (single rate three colour marker) based policers deployed at the border of the CESNET2 MPLS DiffServ domain to ensure required operational policing limits.

For TCP traffic with unknown characteristics the following initial parameter settings are recommended:

- IBE [Byte] = 2 × IBC [Byte].

3.3.3 Classification and Marking

The implementation of QoS on the CESNET2 MPLS-based backbone uses wherever possible a DSCP to Exp value mapping that corresponds to the Cisco IOS implicit MPLS ingress behaviour: the first three bits of the DSCP are copied to the Exp field during MPLS label imposition.

Table 7 shows the CESNET2 MPLS DiffServ domain compliant DSCP mapping for the particular supported service classes.

It is the responsibility of the neighbouring DiffServ domain to ensure proper marking of the IPv4/IPv6 packets entering the CESNET2 MPLS DiffServ domain to be compliant with the CESNET2 QoS services. Owing to the “Short-Pipe” MPLS DiffServ tunnelling mode the DSCP transparency of the in-profile
Table 7. CESNET2 compliant DSCP/Exp/CoS marking for service classes.

<table>
<thead>
<tr>
<th>Service class</th>
<th>PHB</th>
<th>DSCP</th>
<th>Exp MPLS</th>
<th>802.1p CoS</th>
</tr>
</thead>
<tbody>
<tr>
<td>real-time</td>
<td>EF</td>
<td>EF, CS5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>network control</td>
<td>AF</td>
<td>CS6, CS7</td>
<td>6 (7)</td>
<td>6 (7)</td>
</tr>
<tr>
<td>video</td>
<td>AF</td>
<td>AF4x, CS4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>critical traffic</td>
<td>AF</td>
<td>AF3x, CS3, AF2x, CS2</td>
<td>3 (2)</td>
<td>3 (2)</td>
</tr>
<tr>
<td>best effort</td>
<td>default</td>
<td>0 (others)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LBE</td>
<td>AF</td>
<td>AF1x, CS1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

user IPv4/IPv6 packets (SLA conformant) is provided while being transported through the CESNET2 MPLS DiffServ domain.

3.3.4 Authentication, Authorization, and Trust

QoS admission control must be applied by means of policing at all ingress points of the CESNET2 MPLS DiffServ domain in the following manner:

— Ingress srTCM policer checks all the incoming traffic within each particular QoS service class against an SLA.

— All the IPv4/IPv6 packets of the conforming traffic of the particular QoS class remain unchanged and the corresponding Exp mapping (determining the PHB within the CESNET2 MPLS DiffServ domain) is derived from their original DSCP values (see Table 7).

— The DSCP of IPv4/IPv6 packets representing the exceeding/violating (out-of-profile) traffic of the particular QoS service class may be changed and the corresponding Exp mapping (determining the PHB within the CESNET2 MPLS DiffServ domain) is derived from the implied policing rules valid for this type of exceeding/violating traffic (see Table 8).

Policing

QoS admission control must be applied by means of policing at all ingress points of the CESNET2 MPLS DiffServ domain to enforce the SLA with other DiffServ domains for each supported service class. Traffic policing (or shaping) is optional at egress points of the CESNET2 MPLS DiffServ domain.

The ingress QoS operational policing limits are enforced by the srTCM based policers. The goal is to reclassify/markdown input user traffic according to Table 8. If ICR/IPR (the total sum of the traffic across all service classes) equals the absolute link capacity an ordinary srTCM policer is sufficient to check the SLS compliance for each particular service class. Otherwise, if ICR/IPR is less than the absolute link capacity, a hierarchical srTCM policer is required: the first level policer checks the conformity of the total traffic (the whole traffic regardless of the
service classes) and the second level policer checks the conformity of each particular service class being provided. Optionally, the same can be applied to EBC, if the egress QoS operational policing limits are required.

Table 8 shows the proposed reclassification policy of the ingress QoS admission control at the edge of the CESNET2 MPLS DiffServ domain. It is strongly recommended that for incoming traffic the Network Control class be checked against the ACL of authorized external sources from the neighbouring QoS domain to allow only authorized control traffic (e.g. BGP and/or MSDP) to transit (or end up inside) the CESNET2 MPLS DiffServ domain, thus avoiding unwanted malicious competition of unauthorized customer traffic within the privileged Network Control class.

<table>
<thead>
<tr>
<th>Service class</th>
<th>Conform (t&lt;IBC_r)</th>
<th>Exceed (IBC_r,t&lt;IBE_r)</th>
<th>Violate (t&gt;IBE_r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>real-time</td>
<td>real-time</td>
<td>best effort</td>
<td>LBE</td>
</tr>
<tr>
<td>network control</td>
<td>network control</td>
<td>best effort</td>
<td>LBE</td>
</tr>
<tr>
<td>video</td>
<td>video</td>
<td>best effort</td>
<td>LBE</td>
</tr>
<tr>
<td>critical traffic</td>
<td>critical traffic</td>
<td>best effort</td>
<td>LBE</td>
</tr>
<tr>
<td>best effort</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>best effort</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

3.4 QoS Implementation on the CESNET2 MPLS-based Backbone

Technical implementation of the proposed CESNET2 QoS Policy Framework heavily depends on the hardware platform: not only on the particular Cisco router model, but also on the particular module in the case of Cisco 7600:

— Cisco 7600 OSR Sup720-3BXL/3CXL
— equipped with OSM3, SIP4 and ES205 modules can use a proper variant of the LLQ/CBWFQ queueing disciplines for EF/AF PHB on output (configured via the Cisco IOS MQC) with WRED if supported,
— equipped with LAN modules6 can use WRR scheduling with a priority queue

3 OSM-1OC48-POS-SL+, OSM-1OC48-POS-SS+, OSM-2+4GE-WAN+
4 7600-SIP-400/SPA-1XOC48POS/RPR
5 7600-ES20-10G3CXL
6 LAN modules typically denote the internal queueing structure of their interfaces by notation of the number of priority (p) and WRR (q) queues, each with the number of configurable WRED thresholds (t). Thus the abbreviation 1p2q2t means: 1 priority queue, 2 WRR queues each with 2 thresholds.
on output (typically 1p7q8t for 10 Gigabit Ethernet, 1p3q8t or 1p2q2t for Gigabit Ethernet and 1p3q1t for Fast Ethernet output interfaces in case of CESNET2 MPLS backbone) with WRED if supported.

— Cisco CRS-1 can use the LLQ/MDRR queueing disciplines for EF/AF PHB on output (configured via the Cisco IOS XR MQC).

Figure 2. Types of QoS interface in the CESNET2 MPLS DiffServ domain.

Figure 2 shows the classification of the types of interfaces of the CESNET2 MPLS Exp-LSP DiffServ domain from the QoS point of view:

— IP-to-IP: IP based QoS interfaces (the external/neighbouring IPv4/IPv6 Diff-Serv domain):
— CE-PE-out: outgoing CE\textsuperscript{7} interface facing to PE,

\textsuperscript{7} External CE of a neighbouring IPv4/IPv6 DiffServ domain.
QoS Design and Implementation in the CESNET2 MPLS-based Backbone

— CE-PE-in: incoming CE interface from PE.
— IP-to-MPLS: IP based QoS interfaces:
— PE-CE-in: incoming PE interface from CE.
— MPLS-to-IP: IP based QoS interfaces:
— PE-CE-out: outgoing PE interface facing to CE.
— MPLS-to-MPLS: MPLS-based QoS interfaces:
— PE-PE-out: outgoing PE interface facing to another PE,
— PE-PE-in: incoming PE interface from another PE,
— PE-P-out: outgoing PE interface facing to P,
— PE-P-in: incoming PE interface from another P,
— P-PE-out: outgoing P interface facing to PE,
— P-PE-in: incoming P interface from PE.

3.4.1 QoS classification

For traffic classification of supported service classes the following IOS/XR class maps can be used:

*IP-to-MPLS IP-based QoS interfaces*

```plaintext
class-map match-any CESNET2-RealTime-DSCP
description Real-Time
match dscp cs5 ef
class-map match-any CESNET2-NetworkControl-DSCP
description Network Control
match dscp cs6 cs7
class-map match-any CESNET2-Video-DSCP
description Video
match dscp cs4 af41 af42 af43
class-map match-any CESNET2-CriticalTraffic-DSCP
description Critical Traffic
match dscp cs2 af21 af22 af23 cs3 af31 af32 af33
class-map match-any CESNET2-LessEffort-DSCP
description Less than Best Effort
match dscp cs1 af11 af12 af13
```

8 When penultimate hop popping occurs, pure IPv4/IPv6 packets can traverse the last link within the MPLS cloud.
9 Penultimate hop popping can occur.
10 Penultimate hop popping can occur.
class-map match-any CESNET2-BestEffort-DSCP
description Best Effort
match dscp 0

**MPLS-to-MPLS or MPLS-to-IP MPLS-based or IP-based QoS interfaces**

class-map match-any CESNET2-RealTime
description Real-Time
match dscp cs5 ef
match mpls experimental topmost 5
class-map match-any CESNET2-NetworkControl
description Network Control
match dscp cs6 cs7
match mpls experimental topmost 6 7
class-map match-any CESNET2-Video
description Video
match dscp cs4 af41 af42 af43
match mpls experimental topmost 4
class-map match-any CESNET2-CriticalTraffic
description Critical Traffic
match dscp cs2 af21 af22 af23 cs3 af31 af32 af33
match mpls experimental topmost 2 3
class-map match-any CESNET2-LessEffort
description Less than Best Effort
match dscp cs1 af11 af12 af13
match mpls experimental topmost 1
class-map match-any CESNET2-BestEffort

description Best Effort
match dscp 0
match mpls experimental topmost 0

### 3.4.2 Cisco 7600 Sup720-3BXL/3XCL QoS Implementation

Hardware QoS acceleration on Cisco 7600 Sup720-3BXL/3CXL running IOS version 12.2(33)SRB2 is configured by

```mls qos```

Configure DSCP or Exp field packet rewrite on outgoing interfaces according to the internal DSCP (necessary for PFC MPLS QoS compatibility):

```mls qos rewrite ip dscp```
The trust state of the incoming physical IP interfaces with the QoS support enabled must be set to DSCP (the trust state of MPLS encapsulated interfaces does not matter\(^\text{11}\), but in case of possible PHP the DSCP trust state is also necessary):

```bash
interface gigabit...
mls qos trust dscp
```

**QoS admission control and reclassification/markdown**

QoS admission control for IP-to-MPLS traffic is configured only on the PE-CE-in input IP interfaces of ingress PE routers in the CESNET2 MPLS DiffServ domain using the hardware based predefined/preconfigured policed DSCP maps. It is important to note that classification (based on the originally received IP header) and marking (done to the internal DSCP) do not distinguish between IP-to-IP traffic and IP-to-MPLS traffic. The commands used to mark IPv4/IPv6 DSCP and mark Exp have the same result as when marking the internal DSCP. The (PFC) internal DSCP values of the out-of-profile policed packets (corresponding to the exceeding/violating portion of incoming traffic) can be changed by global internal PFC maps. This results in a possible real DSCP packet rewrite on non-MPLS output IP interfaces of the same PE router; however, the Exp value of the newly imposed MPLS header on MPLS output interfaces for all in-profile and out-of-profile packets (corresponding to the dscp-exp map) is derived from the internal (PFC) DSCP which reflects the result of input policing:

```bash
!
! Redefine policed-dscp maps for input admission control
!
! Exceed traffic: normal-burst policed-dscp map
no mls qos map policed-dscp normal-burst
! Reclassify/remark RealTime (EF, CS5) DSCP to BestEffort (0)
mls qos map policed-dscp normal-burst 40 46 to 00
! Reclassify/remark NetworkControl (CS6, CS7) DSCP to BestEffort (0)
mls qos map policed-dscp normal-burst 48 56 to 00
! Reclassify/remark Video (CS4, AF4x) DSCP to BestEffort (0)
mls qos map policed-dscp normal-burst 32 34 36 38 to 00
! Reclassify/remark CriticalTraffic (CS3, AF3x, CS2, AF2x) DSCP
to BestEffort (0)
mls qos map policed-dscp normal-burst 24 26 28 30 to 00
mls qos map policed-dscp normal-burst 16 18 20 22 to 00
!
```

\(^{11}\) Unless explicitly using `no mls qos mpls trust experimental`. 
Ingress IP4/IPv6 admission control and reclassification/markdown example for a 10 Gigabit Ethernet PE-CE-in input IP interface on a PE router is as follows:

```plaintext
! QoS ingress admission control (LAN ports)
! -----------------------------
!
! QoS ingress admission control policy definition from MAN/university/Inet
! - 10 Gigabit Ethernet LAN ports
!
policy-map CESNET2-IP2MPLS-CE2PE-10GE-in
  ! bc(d)=CIR*d/8, be(d)=CIR*d/8
  class CESNET2-NetworkControl-DSCP
  ! CIR: 10Mbps, bc: 500ms, be: 1000ms
  police cir 10000000 bc 625000 be 1250000
  conform-action set-mpls-exp-transmit 6
  exceed-action policed-dscp-transmit
  violate-action policed-dscp-transmit
  class CESNET2-RealTime-DSCP
  ! CIR: 40Mbps, bc(d=125ms)=CIR*d/8=40000000*0.125/8=625000,
  ! be(d=250ms)=CIR*d/8=40000000*0.250/8=1250000
  police cir 40000000 bc 625000 be 1250000
  conform-action set-mpls-exp-transmit 5
  exceed-action policed-dscp-transmit
  violate-action policed-dscp-transmit
  class CESNET2-Video-DSCP
  ! CIR: 60Mbps, bc: 250ms, be: 500ms
  police cir 60000000 bc 1875000 be 3750000
  conform-action set-mpls-exp-transmit 4
  exceed-action policed-dscp-transmit
  violate-action policed-dscp-transmit
```
class CESNET2-CriticalTraffic-DSCP
  ! CIR: 100Mbps, bc: 500ms, be: 1000ms
police cir 1000000000 bc 625000000 be 125000000
  conform-action set-mpls-exp-transmit 3
  exceed-action policed-dscp-transmit
  violate-action policed-dscp-transmit
class CESNET2-LessEffort
set mpls experimental imposition 1
class CESNET2-BestEffort
set mpls experimental imposition 0
class class-default
set mpls experimental imposition 0
!
! PE-CE-in physical ingress admission control for port-based interface
! from MAN/university/Inet
!
interface TenGigabitEthernet ...
  mls qos trust dscp
  ...
  service-policy input CESNET2-IP2MPLS-CE2PE-10GE-in

Implementation of DiffServ EF/AF PHB on ES20, OSM, and SIP/SPA modules

The required DiffServ EF/AF PHB is configured on the output interfaces of the
ES20 modules using the LLQ/CBWFQ queueing disciplines via Cisco IOS MQC
in the following manner:

! ES20 modules output QoS policy definition (IP/MPLS), WRED not supported
! ---------------------------------------------
! 7600-ES20-10G3CXL
!
policy-map CESNET2-DiffServ-ES20-out
  class CESNET2-RealTime
  priority
    ! CIR: 2.5Gbps (bc, be: not supported on ES20)
police 2500000000 conform-action transmit exceed-action drop
  class CESNET2-NetworkControl
  bandwidth percent 2
  class CESNET2-Video
  bandwidth percent 20
  class CESNET2-CriticalTraffic
  bandwidth percent 25
  class CESNET2-LessEffort
  bandwidth percent 3
The required DiffServ EF/AF PHB is configured on output interfaces of the OSM/SIP/SPA modules using the LLQ/CBWFQ queueing disciplines via Cisco IOS MQC in the following way:

```
! OSM/SIP/SPA PoS OC48 modules output QoS policy definition (IP/MPLS)
! -------------------------------------------------------------------
! OSM-1OC48-POS-SL+, OSM-1OC48-POS-SS+, 7600-SIP-400/SPA-1XOC48POS/RPR
!
policy-map CESNET2-DiffServ-OSM-POS-out
  class CESNET2-RealTime
    priority percent 25
  class CESNET2-NetworkControl
    bandwidth percent 2
    random-detect dscp-based
  class CESNET2-Video
    bandwidth percent 20
  class CESNET2-CriticalTraffic
    bandwidth percent 25
    random-detect dscp-based
  class CESNET2-LessEffort
    bandwidth percent 3
    random-detect dscp-based
  class class-default
    bandwidth percent 25
    random-detect dscp-based
```

The required DiffServ EF/AF PHB is configured on output interfaces of the OSM GE-WAN modules using the LLQ/CBWFQ queueing disciplines via Cisco IOS MQC as follows:

```
! GE-WAN modules output QoS policy definition (IP/MPLS)
! -----------------------------------------------------
! OSM-2+4GE-WAN+
!
policy-map CESNET2-DiffServ-OSM-GEWAN-out
  class CESNET2-RealTime
  priority
    ! CIR: 250Mbps, bc(d=125ms)=CIR*d/8=250000000*0.125/8=3906250,
```
! be(d=250ms) = CIR*d/8 = 250000000*0.250/8 = 7812500
police cir 250000000 bc 3906250
  conform-action transmit
  exceed-action drop
  class CESNET2-NetworkControl
  bandwidth percent 2
  random-detect dscp-based
  class CESNET2-Video
  bandwidth percent 20
  class CESNET2-CriticalTraffic
  bandwidth percent 25
  random-detect dscp-based
  class CESNET2-LessEffort
  bandwidth percent 3
  random-detect dscp-based
  class class-default
  bandwidth percent 25
  random-detect dscp-based

! OSM modules output QoS policy configuration (IP/MPLS)
! -----------------------------------------------------
!
! Physical routed GE/WAN MPLS interface
interface GE-WAN ...
mls qos trust dscp
service-policy output CESNET2-DiffServ-OSM-GEWAN-out

Implementation of the DiffServ EF/AF PHB on LAN modules
Tables 9, 10, 11 and 12 show the initial DiffServ EF/AF PHB queue scheduling and Exp/CoS mapping for all 1p7q8t, 1p3q8t, 1p2q2t, and 1p3q1t output ports of the LAN modules (Exp/CoS=Tx denotes the Exp/CoS values to the particular threshold number assignment, Tx denotes the threshold number).

The required DiffServ EF/AF PHB is configured on all IP and MPLS output interfaces of the LAN modules using the following WRR/priority queues specific commands:

1p7q8t LAN 10 Gigabit Ethernet output ports
!
! 1p7q8t LAN 10GE output ports
! --------------------
Table 9. The IP/MPLS DiffServ compliant output queueing for 1p7q8t LAN 10GE ports. Abbreviations: Q=Queue, W=Weight, QL=Queue-limit.

<table>
<thead>
<tr>
<th>Q Type</th>
<th>Exp/CoS=Tx</th>
<th>W</th>
<th>QL</th>
<th>T1min</th>
<th>T1max</th>
<th>T2-8min</th>
<th>T2-8max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 WRR 1=T1</td>
<td>3</td>
<td>5%</td>
<td>80%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>2 WRR 0=T1</td>
<td>25</td>
<td>25%</td>
<td>80%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>3 WRR 2=T1, 3=T1</td>
<td>25</td>
<td>25%</td>
<td>80%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>4 WRR 4=T1</td>
<td>20</td>
<td>15%</td>
<td>80%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>5 WRR 6=T1, 7=T1</td>
<td>2</td>
<td>5%</td>
<td>80%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>6 WRR —</td>
<td>—</td>
<td>—</td>
<td>80%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>7 WRR —</td>
<td>—</td>
<td>—</td>
<td>80%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>8 Priority 5</td>
<td>—</td>
<td>25%</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 10. The IP/MPLS DiffServ compliant output queueing for 1p3q8t LAN 10GE ports. Abbreviations: Q=Queue, W=Weight, QL=Queue-limit.

<table>
<thead>
<tr>
<th>Q Type</th>
<th>Exp/CoS=Tx</th>
<th>W</th>
<th>QL</th>
<th>T1min</th>
<th>T1max</th>
<th>T2min</th>
<th>T2max</th>
<th>T3min</th>
<th>T3max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 WRR 1=T1</td>
<td>3</td>
<td>5%</td>
<td>0%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>2 WRR 0=T1</td>
<td>25</td>
<td>25%</td>
<td>0%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>3 WRR 4=T1, 2=T2, 3=T2, 6=T3</td>
<td>72</td>
<td>45%</td>
<td>70%</td>
<td>80%</td>
<td>80%</td>
<td>90%</td>
<td>90%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>4 Priority 5</td>
<td>—</td>
<td>25%</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 11. The IP/MPLS DiffServ compliant output queueing for 1p2q2t LAN GE ports. Abbreviations: Q=Queue, W=Weight, QL=Queue-limit.

<table>
<thead>
<tr>
<th>Q Type</th>
<th>Exp/CoS=Tx</th>
<th>W</th>
<th>QL</th>
<th>T1min</th>
<th>T1max</th>
<th>T2min</th>
<th>T2max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 WRR low 1=T1, 0=T2</td>
<td>25</td>
<td>40%</td>
<td>40%</td>
<td>80%</td>
<td>80%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>2 WRR high 2=T1, 3=T1, 4=T1, 6=T2, 7=T2</td>
<td>75</td>
<td>30%</td>
<td>70%</td>
<td>90%</td>
<td>90%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>3 Priority 5</td>
<td>—</td>
<td>30%</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 12. The IP/MPLS DiffServ compliant output queueing for 1p3q1t LAN FE ports. Abbreviations: Q=Queue, W=Weight, QL=Queue-limit.

<table>
<thead>
<tr>
<th>Q Type</th>
<th>Exp/CoS=Tx</th>
<th>W</th>
<th>QL</th>
<th>T1min</th>
<th>T1max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 WRR 1=T1</td>
<td>3</td>
<td>80%</td>
<td>100%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 WRR 0=T1</td>
<td>25</td>
<td>80%</td>
<td>100%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 WRR 2=T1, 3=T1, 4=T1, 6=T1, 7=T1</td>
<td>72</td>
<td>90%</td>
<td>100%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Priority 5</td>
<td>—</td>
<td>0%</td>
<td>0%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
QoS Design and Implementation in the CESNET2 MPLS-based Backbone

! WS-X6704-10GE
!
! PQ: RealTime25%EF, CS5Exp5Cos5
! QST1: NetWorkControl 2%CS7, CS6Exp6(7)Cos6(7)
! Q4T1: Video20%CS4, AF4xExp4Cos4
! Q3T1: CriticalTraffic25%CS3, AF3x, CS2, AF2xExp3(2)Cos3(2)
! Q2T1: BestEffort25%0Exp0Cos0
! Q1T1: LessEffort3%CS1, AF1xExp1Cos1
!
!----------------------------------------------------------------------
!
!>>> interface Ten1/4
!
default wrr-queue bandwidth

default wrr-queue cos-map

default wrr-queue queue-limit

default wrr-queue random-detect 1

default wrr-queue random-detect 2

default wrr-queue random-detect 3

default wrr-queue random-detect 4

default wrr-queue random-detect 5

default wrr-queue random-detect 6

default wrr-queue random-detect 7

default wrr-queue threshold 1

default wrr-queue threshold 2

default wrr-queue threshold 3

default wrr-queue threshold 4

default wrr-queue threshold 5

default wrr-queue threshold 6

default wrr-queue threshold 7
!
! size: q1% q2% q3% q4% q5% q6% q7%
!

wrr-queue queue-limit 5 25 25 15 5 0 0
! Allocates 5% to Q1, 25% to Q2, 25% to Q3, 15% to Q4,
! Allocates 5% to Q5, 0% to Q6 and 0% to Q7
!

wrr-queue bandwidth 3 25 25 20 2 0 0
! Sets the WRR weights for 3:25:25:20:2:0:0 (Q1 through Q7)
!

wrr-queue random-detect 1
wrr-queue random-detect 2
wrr-queue random-detect 3
wrr-queue random-detect 5
!

wrr-queue random-detect 1
wrr-queue random-detect 2
wrr-queue random-detect 3
wrr-queue random-detect 5
!

wrr-queue random-detect 1
wrr-queue random-detect 2
wrr-queue random-detect 3
wrr-queue random-detect 5
!

wrr-queue threshold 1
wrr-queue threshold 2
wrr-queue threshold 3
wrr-queue threshold 4
wrr-queue threshold 5
wrr-queue threshold 6
wrr-queue threshold 7
!

wrr-queue threshold 1
wrr-queue threshold 2
wrr-queue threshold 3
wrr-queue threshold 4
wrr-queue threshold 5
wrr-queue threshold 6
wrr-queue threshold 7
!

wrr-queue threshold 1
wrr-queue threshold 2
wrr-queue threshold 3
wrr-queue threshold 4
wrr-queue threshold 5
wrr-queue threshold 6
wrr-queue threshold 7
!

wrr-queue threshold 1
wrr-queue threshold 2
wrr-queue threshold 3
wrr-queue threshold 4
wrr-queue threshold 5
wrr-queue threshold 6
wrr-queue threshold 7
!

wrr-queue threshold 1
wrr-queue threshold 2
wrr-queue threshold 3
wrr-queue threshold 4
wrr-queue threshold 5
wrr-queue threshold 6
wrr-queue threshold 7
!
wrr-queue random-detect min-threshold 1 80 100 100 100 100 100 100
wrr-queue random-detect min-threshold 2 80 100 100 100 100 100 100
wrr-queue random-detect min-threshold 3 80 100 100 100 100 100 100
wrr-queue random-detect min-threshold 4 80 100 100 100 100 100 100
wrr-queue random-detect min-threshold 5 80 100 100 100 100 100 100
wrr-queue random-detect min-threshold 6 80 100 100 100 100 100 100
wrr-queue random-detect min-threshold 7 80 100 100 100 100 100 100
wrr-queue random-detect max-threshold 1 100 100 100 100 100 100 100
wrr-queue random-detect max-threshold 2 100 100 100 100 100 100 100
wrr-queue random-detect max-threshold 3 100 100 100 100 100 100 100
wrr-queue random-detect max-threshold 4 100 100 100 100 100 100 100
wrr-queue random-detect max-threshold 5 100 100 100 100 100 100 100
wrr-queue random-detect max-threshold 6 100 100 100 100 100 100 100
wrr-queue random-detect max-threshold 7 100 100 100 100 100 100 100
! q: queue# threshlod# cos#

wrr-queue cos-map 1 1 1
wrr-queue cos-map 2 1 0
wrr-queue cos-map 3 1 2 3
wrr-queue cos-map 4 1 4
wrr-queue cos-map 5 1 6 7
! p: queue# cos#

priority-queue cos-map 1 5

1p3q8t LAN Gigabit Ethernet output ports
!

! 1p3q8t LAN GE output ports
! --------------------------
! WS-X6724-5FP, WS-X6748-GE-TX
!
! PQ: RealTime25%EF, CS5Exp5Cos5
! Q3T3: NetWorkControl 2%CS7, CS6Exp6(7)Cos6(7)
! Q3T1: Video20%CS4, AF4xExp4Cos4
! Q3T2: CriticalTraffic25%CS3, AF3x, CS2, AF2xExp3(2)Cos3(2)
! Q2T1: BestEffort25%Exp0Cos0
! Q1T1: LessEffort3%CS1, AF1xExp1Cos1
!
!---------------------------------------------------------------------
!
!>>> interface Giga4/4

default wrr-queue bandwidth
default wrr-queue cos-map
default wrr-queue queue-limit
default wrq-queue random-detect 1
default wrq-queue random-detect 2
default wrq-queue random-detect 3
default wrq-queue threshold 1
default wrq-queue threshold 2
default wrq-queue threshold 3
! size: q1% q2% q3%
wrr-queue queue-limit 5 25 45
! weight: q1 q2 q3
  wrq-queue bandwidth 3 25 72
! enable WRED: q1, q2, q3
wrr-queue random-detect 1
wrr-queue random-detect 2
wrr-queue random-detect 3
! threshold: q# t1% t2% t3% t4% t5% t6% t7% t8%
wrr-queue random-detect min-threshold 1 80 100 100 100 100 100 100 100
wrr-queue random-detect min-threshold 2 80 100 100 100 100 100 100 100 100
wrr-queue random-detect min-threshold 3 70 80 90 100 100 100 100 100 100
wrr-queue random-detect max-threshold 1 100 100 100 100 100 100 100 100 100
wrr-queue random-detect max-threshold 2 100 100 100 100 100 100 100 100 100
wrr-queue random-detect max-threshold 3 80 90 100 100 100 100 100 100 100 100
! p: queue# threshlod# cos#...
wrr-queue cos-map 1 1 1
wrr-queue cos-map 2 1 0
wrr-queue cos-map 3 1 4
wrr-queue cos-map 3 2 2 3
wrr-queue cos-map 3 3 6 7
! p: queue# cos#...
priority-queue cos-map 1 5
1p2q2t LAN Gigabit Ethernet output ports

! 1p2q2t LAN GE output ports
! --------------------------
! WS-SUP720-3BXL, WS-X6516-GBIC, WS-X6516A-GBIC, WS-X6816-GBIC,
! WS-X6548-GE-TX,
! OSM-1OC48-POS-5L+, OSM-1OC48-POS-5S+, OSM-2+4GE-WAN+
! PQ: RealTime25%EF, CS5Exp5Cos5
! Q2T2: NetWorkControl 2%CS7, CS6Exp6(7)Cos6(7)
! Q2T1: Video20%CS4, AF4xExp4Cos4
! Q2T1: CriticalTraffic25%CS3, AF3x, CS2, AF2xExp3(2)Cos3(2)
! Q1T2: BestEffort25%Exp0Cos0
! Q1T1: LessEffort3%CS1, AF1xExp1Cos1
!

---------------------------------------------------------------------

>>> interface Giga4/4
default wrr-queue bandwidth
default wrr-queue cos-map
default wrr-queue queue-limit
! size: q1% q2%[=p1%]
wrr-queue queue-limit 40 30
! weight: q1 q2
wrr-queue bandwidth 25 75
! threshold: q# t1% t2%
wrr-queue random-detect min-threshold 1 40 80
wrr-queue random-detect min-threshold 2 70 90
wrr-queue random-detect max-threshold 1 80 100
wrr-queue random-detect max-threshold 2 90 100
! q: queue# threshold# cos#...
wrr-queue cos-map 1 1 1
wrr-queue cos-map 1 2 0
wrr-queue cos-map 2 1 2 3 4
wrr-queue cos-map 2 2 6 7
! p: queue# cos#...
priority-queue cos-map 1 5
1p3q1t LAN Fast Ethernet output ports

1p3q1t LAN FE output ports

- WS-X6548-RJ-45

- PQ: RealTime25%EF, CS5Exp5Cos5
- Q3T1: NetworkControl 2%CS7, CS6Exp6(7)Cos6(7)
- Q3T1: Video20%CS4, AF4xExp4Cos4
- Q3T1: CriticalTraffic25%CS3, AF3x, CS2, AF2xExp3(2)Cos3(2)
- Q2T1: BestEffort25%Exp0Cos0
- Q1T1: LessEffort3%CS1, AF1xExp1Cos1

----------------------------------------------------------------------

>>> interface Fast8/3
default wrr-queue bandwidth
default wrr-queue cos-map
! weight: q1 q2 q3
wrr-queue bandwidth 3 25 72
! threshold: q# t1%
wrr-queue random-detect min-threshold 1 80
wrr-queue random-detect min-threshold 2 80
wrr-queue random-detect min-threshold 3 90
wrr-queue random-detect max-threshold 1 100
wrr-queue random-detect max-threshold 2 100
wrr-queue random-detect max-threshold 3 100
! q: queue# threshold# cos#
wrr-queue cos-map 1 1 1
wrr-queue cos-map 2 1 0
wrr-queue cos-map 3 1 2 3 4 6 7
! p: queue# cos#
priority-queue cos-map 1 5

3.4.3 Cisco CRS-1 QoS Implementation

QoS admission control and reclassification/markdown

QoS admission control for IP-to-MPLS traffic is configured only on the PE-CE-in input IP interfaces of ingress PE routers into the CESNET2 MPLS DiffServ domain using the Cisco IOS XR MQC in the following way:
Ingress IP4/IPv6 admission control and reclassification/markdown example for a Gigabit Ethernet PE-CE-in input IP interface on a PE router is as follows:

```plaintext
! QoS ingress admission control policy definition from MAN/university/Inet
! - Gigabit Ethernet LAN ports
!
policy-map CESNET2-IP2MPLS-CE2PE-GE-in
  ! bc(d)=CIR*d/8, be(d)=CIR*d/8
  class CESNET2-NetworkControl-DSCP
  ! CIR: 10Mbps, bc: 500ms, be: 1000ms
  police cir 10000000 bc 625000 be 1250000
    conform-action set mpls experimental imposition 6
    exceed-action set mpls experimental imposition 0
    violate-action set mpls experimental imposition 1
  class CESNET2-RealTime-DSCP
  ! CIR: 40Mbps, bc(d=125ms)=CIR*d/8=40000000*0.125/8=625000, be(d=250ms)=CIR*d/8=40000000*0.250/8=1250000
  police cir 40000000 bc 625000 be 1250000
    conform-action set mpls experimental imposition 5
    exceed-action set mpls experimental imposition 0
    violate-action set mpls experimental imposition 1
  class CESNET2-Video-DSCP
  ! CIR: 60Mbps, bc: 250ms, be: 500ms
  police cir 60000000 bc 1875000 be 3750000
    conform-action set mpls experimental imposition 4
    exceed-action set mpls experimental imposition 0
    violate-action set mpls experimental imposition 1
  class CESNET2-CriticalTraffic-DSCP
  ! CIR: 50Mbps, bc: 500ms, be: 1000ms
  police cir 50000000 bc 3125000 be 6250000
    conform-action set mpls experimental imposition 3
    exceed-action set mpls experimental imposition 0
    violate-action set mpls experimental imposition 1
  class CESNET2-LessEffort
  set mpls experimental imposition 1
  class CESNET2-BestEffort
  set mpls experimental imposition 0
  class class-default
  set mpls experimental imposition 0

Ingress IP4/IPv6 admission control and reclassification/markdown example for a Gigabit Ethernet PE-CE-in input IP interface on a PE router is as follows:

! PE-CE-in physical ingress admission control for port-based interface from
! MAN/university/Inet
```
interface GigabitEthernet ... 
  service-policy input CESNET2-IP2MPLS-CE2PE-GE-in

Implementation of the DiffServ EF/AF PHB

The required DiffServ EF/AF PHB is configured on output interfaces using the LLQ/MDRR queueing disciplines via Cisco IOS XR MQC in the following way:

! 
! Output backbone policy (MPLS/IP) 
! ===================================
! 
! policy-map CESNET2-DiffServ-out 
  class CESNET2-RealTime
      ! CIR: 2.5Gbps, bc(d=125ms)=CIR*d/8=2500000000*0.125/8=39062500, 
      ! be(d=250ms)=CIR*d/8=2500000000*0.250/8=78125000
      police rate percent 25 burst 125 ms peak-burst 250 ms 
      conform-action transmit 
      exceed-action set mpls experimental topmost 0 
      violate-action drop 
      ! priority 
      !
  class CESNET2-NetworkControl 
      bandwidth percent 2 
      random-detect default 
      !
  class CESNET2-Video 
      bandwidth percent 20 
      !
  class CESNET2-CriticalTraffic 
      bandwidth percent 25 
      random-detect default 
      !
  class CESNET2-LessEffort 
      bandwidth percent 3 
      random-detect default 
      !
  class class-default 
      bandwidth percent 25 
      random-detect default 
      !
end-policy-map
! Output QoS policy configuration (IP/MPLS)
!-----------------------------------------
!
! Physical routed GE MPLS interface
  interface GigabitEthernet ...
  service-policy output CESNET2-DiffServ-out

4 Conclusion
Currently there is only a little need for QoS coming from general academic users. This can be justified by the fact that CESNET2 MPLS-based backbone operational behaviour is based on an over-provisioned network approach for common user traffic. On the other hand we believe that some emerging extensive applications like real-time data acquisition from remote sensors, high definition video and GRID computation will need delay, jitter and packet loss guarantees in the near future no matter what the increase of common user traffic will be.

It is good to remember that QoS introduces a system of managed unfairness. The successful QoS policy rollout should be followed by ongoing monitoring of service levels and periodic adjustments and tuning of QoS policies. As traffic conditions change, we will have to adapt to these changes and maybe start the QoS deployment cycle anew, by redefining the objectives, tuning and testing corresponding designs, rolling out new designs and monitoring them to check if they match the redefined objectives.

The major drawback of the proposed CESNET2 QoS policy implementation is the fact that real QoS PHB on egress IPv4/IPv6 edge interfaces of the CESNET2 MPLS DiffServ domain towards other neighbouring QoS domains may be based on the original DSCP of user packets entering the CESNET2 MPLS DiffServ domain (due to the “Short-Pipe” MPLS DiffServ tunnelling mode because of the possibility of penultimate hop popping). This means that the real egress traffic mix may not obey the ingress QoS admission control rules. A question remains whether it is worth another limited output QoS policing to enforce the egress QoS admission rules once the user packets have already been successfully transported throughout the CESNET2 MPLS DiffServ domain or whether they should rather obey the (usually mandatory) ingress QoS admission control rules when entering the neighbouring QoS domain. Penultimate hop popping (in the case of not using MPLS/BGP VPN technology, when only one MPLS label is present) can be avoided by using the mpls 1dp explicit-null global configuration command, but unfortunately in case of Cisco 7600 Sup720-3BXL/3CXL PFC QoS only at the cost of possible drastic forwarding performance degradation (by 50%).
5 Glossary of Terms

- AF – Assured Forwarding
- AF-PHB – Assured Forwarding Per Hop Behaviour
- BGP – Border Gateway Protocol
- CE – Customer Equipment (router)
- CBR – Constant Bit Rate
- CBWFQ – Class-Based Weighted Fair Queueing
- CoS – Class of Service (IEEE 802.1p bits)
- DiffServ – Differentiated Services
- DS – Differentiated Services
- DSCP – Differentiated Services Code Point
- EBC – Egress Burst Conform for the whole traffic (regardless QoS classes)
- EBC\textsuperscript{class} – Egress Burst Conform within the class
- EBE – Egress Burst Exceed for the whole traffic (regardless QoS classes)
- EBE\textsuperscript{class} – Egress Burst Exceed within the class
- ECR – Egress Committed Rate for the whole traffic (regardless QoS classes)
- ECR\textsuperscript{class} – Egress Committed Rate within the class
- EF – Expedited Forwarding
- EF-PHB – Expedited Forwarding Per Hop Behaviour
- EPR – Egress Peek Rate for the whole traffic (regardless QoS classes)
- EPR\textsuperscript{class} – Egress Peek Rate within the class
- Exp – Experimental field of the MPLS header
- Exp-LSP – Label Switch Path based on the Experimental field
- E-LSP – Label Switch Path based on the Experimental field
- IBC – Ingress Burst Conform for the whole traffic (regardless QoS classes)
- IBC\textsuperscript{class} – Ingress Burst Conform within the class
- IBE – Ingress Burst Exceed for the whole traffic (regardless QoS classes)
- IBE\textsuperscript{class} – Ingress Burst Exceed within the class
- ICR – Ingress Committed Rate for the whole traffic (regardless QoS classes)
- ICR\textsuperscript{class} – Ingress Committed Rate within the class
- IETF – Internet Engineering Task Force
- IP – Internet Protocol
- IPv4 – Internet Protocol version 4
- IPv6 – Internet Protocol version 6
- IPR – Ingress Peek Rate for the whole traffic (regardless QoS classes)
- IPR\textsuperscript{class} – Ingress Peek Rate within the class
- LAN – Local Area Network
- LBE – Less than Best Efforts
- LLQ – Low Latency Queueing
— MAN – Metropolitan Area Network
— MBS – Managed Bandwidth Service
— MIB – Management Information Base
— MPLS – Multiple Protocol Label Switching
— MQC – Modular Quality of Service Command-Line Interface
— MDRR – Modified Deficit Round Robin
— NREN – National Research and Education Network
— OSM – Optical Service Module
— P – Provider (router)
— PE – Provider Edge (router)
— PHB – Per Hop Behaviour
— PHP – Penultimate Hop Popping
— PFC – Policy Feature Card
— PIP – Premium IP
— PoP – Point of Presence
— PoS – Packet over SONET/SDH
— QoS – Quality of Service
— RED – Random Early Detection
— SLA – Service Level Agreement
— SLS – Service Level Specification
— sr+bs – General policing mechanism that provides single rate with burst size control
— srTCM – single rate Three Colour Marker [10]
— trTCM – two-rate Three Colour Marker [10]
— TE – Traffic Engineering
— VLAN – Virtual Local Area Network
— VPN – Virtual Private Network
— WRED – Weighted Random Early Detection
— WRR – Weighted Round Robin

References


[13] *Quality of Service on GÉANT2*. Available online[^21].


[^15]: http://www.dante.net/sequin/deliverables/SEQ-01-043.pdf
[^21]: http://www.geant2.net/server/show/conWebDoc.1582
[^22]: http://archive.dante.net/sequin/deliverables/SEQ-02-010.pdf
Part II

Network Monitoring and Security
Is Spam Visible in Flow-Level Statistics?

MARTIN ŽÁDNÍK, ZBYNĚK MICHALOVSKÝ

Abstract
This paper investigates the feasibility of detection of spam connections using flow statistics collected upon SMTP connections only. To this end, the paper analyzes several days of SMTP communication collected at a medium-sized email server. In order to prove that spam connections can be automatically identified at the TCP/IP layer we utilized a supervised learning algorithm to construct the classifier, in our case the decision tree. The quality of the classifier is evaluated and results show that the flow based statistics contain a detectable fingerprint specific to spam connections. Such findings may help with further study of spam behaviour in a broader manner as the flow statistics can be collected on-line at the backbone links where it is possible to see SMTP traffic for more than one email server.

Keywords: network measurement, spam, identification, characteristics

1 Introduction
There are many methods and anti-spam techniques of filtering unsolicited mail. One of the most widely used and the most effective method is DNS blacklisting (DNSBL), which filters incoming mail on the basis of identifying the spammer's IP address. The majority of mail servers are configured to reject or flag email that has been sent from an IP that is listed in one of many DNSBL-databases. Other techniques are based on recognizing the pattern or regular expression in email messages. Other methods rely on strict adherence of RFC, e.g. helo/ehlo checking, graylisting, etc. Additionally, very popular are filters that include statistical methods like Bayes filters [6] or elements of artificial intelligence like neural networks. Less common approaches for filtering spam are based on comparison between the receiving CRC email and a database of the CRC’s junk messages, such as Distributed Checksum Clearinghouses1.

A popular way of spreading spam is to use botnets. It is a group of remotely controlled computers compromised by a hacker, computer virus or Trojan. Bots do not send spam in bulks, which makes it difficult for Internet service providers (ISP) to separate spam-traffic from regular traffic at a glance. On the other hand ISPs are able to filter all outgoing SMTP communication coming from open relay

1 http://www.rhyolite.com/anti-spam/dcc/
servers and thus eliminate the possibility of being a source of spam themselves. Unfortunately it is rarely used. Another approach to filtering unsolicited emails coming from spam relay machines is described in [7]. The authors suggest detecting excessive numbers of SMTP connections established by hosts on the monitored network segment. The spam filtering method presented in [8] is based on classification of mail delivery traffic into different categories by the similarity of message contents. If the number of similar mails in any category exceeds a spam threshold, then this category is marked as spam. In contrast with these methods, our approach is focused on identifying SPAM-flow directly on the network layer.

In a similar way, the authors of [5] studied the network-level behaviour of the spam traffic, including: IP address ranges that send the most spam, common spamming modes (e.g. BGP route hijacking, bots), how persistent across time each spamming host is, and characteristics of spamming botnets. But they do not aspire to anatomize the flow characteristics in depth.

Our work was inspired by the successful utilization of traffic flow characteristics to classify network traffic into categories [3] or to reveal the traffic of specific applications, for example to detect VoIP traffic [1]. In this context, we presumed that spammers utilize dedicated SMTP engines to spread spam and these engines differ from other SMTP implementations. Therefore it should be possible to observe them in statistics such as TCP window size or packet size, inter-packet intervals, etc.

This paper contributes to the current research of spam filtering and analysis of spam behaviour. It provides evidence that it is possible to detect spam based on statistical tracking TCP/IP flow characteristics rather than to inspect content of the connection itself. The goal of this presented approach is not to filter 100% of junk email messages. Its main contribution is in spam detection at the level of network backbone traffic where it is possible to observe the behaviour of spamming hosts in a broader context as well as provide additional information in order to support spam-blocking techniques.

The structure of the text is organized as follows: first, in section 2 we present the setup of traffic measurement and annotation architecture. Next, section 3 gives a short overview about the decision tree classifier used for automated classification of SMTP traffic. Collected data and results of classification are analyzed in section 4. Finally, section 5 briefly summarizes the paper and outlines our future work.
2 Experimental Setup

We collected data from the SMTP server hosting mailboxes of the Liberouter\textsuperscript{2} project group. The online service DNS-Blacklisting was switched on, thus allowed us to obtain the part of SMTP flow where the connection from blacklisted IP is refused. We presumed that the majority of SMTP servers use DNS-Blacklisting as an effective counter measure against spam. Emails that made it through the DNS-Blacklisting were delivered into users’ mailboxes and at the same time were stored for further offline analysis.

In parallel, all SMTP and SMTPS traffic was dumped in the file. The file was processed later on by a script that measured 30 unidirectional flow characteristics in each direction per each connection. We have chosen only characteristics that are potentially feasible to measure online over high-speed network traffic. The following constraints were applied:

1. one pass-through data set to enumerate the characteristic
2. a small amount of memory (max. 8 B) to store the item characteristic during the measurement of the flow.

For example maximum, minimum, average, and variance were traced for the length of the packet, interval between consequent packets of the same flow, TCP window size, and other values (a subset of discriminators presented by Moore in \cite{moore}).

In offline mode, the delivered mails were classified by SpamAssassin\textsuperscript{3} version 3.2.3 into two groups: relevant emails and spam. For mail filtering were used these components: Bayes System (with auto-learning); Network test as RBL checks, Razor2, DCC and Pyzor; etc. Czech and English were set as accepted languages. Additionally, we left the threshold score on a default value of 5.0. Further, we applied the configuration tool SpamAssassin Configuration\textsuperscript{4} for the basic settings and then adjusted the configuration manually (e.g. learning Bayes). After the classification of these emails, it was possible to assign a label to each flow which specified the type of the network communication. It could be classified as a relevant email, spam or other communication (attack, outgoing email, failed communication). By using these actions and data modification we obtained data that were ready for classification 2.

\textsuperscript{2} \url{http://www.liberouter.org}
\textsuperscript{3} \url{http://spamassassin.apache.org/}
\textsuperscript{4} \url{http://www.yrex.com/spam/spamconfig.php}
3 Classification

Our intention was to use a suitable classification method to validate that spam can be detected in statistics collected upon an SMTP connection. The classification method should satisfy the following criteria:

1. Low cost of evaluation
2. Accurate results
3. Learning process including the feature selection
4. Easy to understand

The key characteristics of every classification method is high accuracy but at the same time the trained classifier must be simple and easy to evaluate. Fast
evaluation was of the greatest importance considering it should run in the device processing multi-gigabit backbone link. On the other hand the training process may be quite complex and it may take several hours to train the classifier.

The essential task was to select the proper features from the feature set which contributed. Some classification methods can deal with feature selection during the learning process, some methods require having the feature set filtered; otherwise the training process does not work well.

Our goal was not to tune the classification but rather to prove its feasibility for spam detection. Therefore we preferred to use the more sophisticated method to train the classifier that can deal with feature selection as well. We chose the decision tree induction approach as the best fit for our requirements.

The decision tree was build to recursively partition the data into smaller subsets until residual instances in the leaves belong to the same class. The following algorithm describes the process of constructing the tree as described in [2].

1. Initialize by setting variable $T$ to be the training set.
2. Apply the following steps to $T$.
   1. If all elements in $T$ are of class $c_j$, create a $c_j$ node and halt.
   2. Otherwise select a feature $F$ with values $v_1, v_2, \ldots, v_N$. Partition $T$ into $T_1, T_2, \ldots, T_N$, according to their values on $F$. Create a branch with $F$ as a parent node and $T_1, T_2, \ldots, T_N$ as child nodes.
   3. Apply the procedure recursively to each child node.

Feature selection is a fundamental part of decision tree induction. It is based on selecting a feature $F$ that minimizes the information $I_F(S)$ necessary to further classify subsets received by partitioning set $S$ using feature $F$. In other words, to maximize the information gain $G(F)$ which is computed as the difference of the entropy of set $S$ and the entropy after partitioning by $F$.

$$G(F) = I(S) - I_F(S),$$

$$I(S) = - \sum_{i=1}^{n} p_i \log_2 p_i,$$

$$I_F(S) = - \sum_{j=1}^{v} \frac{|S_j|}{|S|} I(S_j).$$

Such an approach allows selection of good features and creation of a simple but not necessarily optimal tree.

During our experiments we used Weka\(^5\) which implements several modifications of decision tree induction algorithm. It is well known that each classification

\(^5\)\url{http://www.cs.waikato.ac.nz/ml/weka/}
algorithm or its modification may perform differently for particular problems. Therefore tests were conducted with available decision tree algorithms and the results of the best performing are presented in Section 4.2. In our case Random Forest performed the best. It constructs several decision trees and the class of the instance is determined by the most votes over all the trees.

4 Results

The results of our measurement are twofold. General properties of measured data are discussed first and then feasibility of flow based statistics for task of spam detection is analyzed.

4.1 Measured Data

The data were collected during 9 days from 07/04/2008 to 16/04/2008 at the email server. The server hosts over one hundred mailboxes for participants of the Liberouter project. Participants are advised to use these addresses carefully, e.g. do not use them as contact emails to register for free service etc.

The raw communication, i.e. raw packets, was dumped with usage of tcpdump. The total size of the file was 235 MB after nine days.

In parallel to tcpdump, received emails were saved to separate folders as described in Section 2. The total number of received messages was 11559 with the size of 278 MB. The log of SMTP communication also took about 270 MB.

Collected messages were annotated by the parsing script which used the log of SMTP communication and the results of SpamAssassin. The annotated data set was manually and randomly checked for correctness; about 2% of emails were annotated incorrectly (mostly weekly reports were identified as spam but it was easy to correct this, the rest was spam that seemed to have a more serious content and was classified as non-spam, i.e. y_spam).

The first point of our interest was to find out whether one spamming host has sent more than one spam to our email server. Surprisingly, most of the spam servers have sent only one spam during the whole measurement period. The average number of accepted spams per host is 1.30 and the average number of rejected spams per host is 1.71 (a histogram is displayed in Figure 3).

Figure 4 outlines the obtained number of spams dependant on IP addresses. It seems that from certain IP ranges like 94.X.X.X to 115.X.X.X there are no received spams at all. The reason might be that these subnets allow to send emails via dedicated gateways and so prevent themselves from being a source of spam. It is important to mention that the source IP addresses cannot be spoofed due to the bidirectional communication required by SMTP for successful message transfer.
Further, we wanted to find out if the amount of received spam is correlated in the time domain, i.e. if the amount of spam forms a pattern that is repeated every day or for any longer period. Because our measurement lasted for only nine days, the longest visible period was necessarily shorter than this time span. The amount of received spam per hour was aggregated and plotted in the graph in Figure 5. One significant outlier emerged on the 14th of April at 6 a.m. with 268 accepted spams. Detailed analysis revealed that our colleague became a victim of email address spoofing, i.e. spammers used his email address and inserted it in the field From: in their spam messages. Most of the target email servers recognized such emails as spam and blocked it. At the same time they replied to our colleague with emails containing the original spam and the message about refusal. Consequently, these emails were recognized correctly as unsolicited email during the annotation process. Such indirect spam was not in the scope of our time analysis and was omitted. Otherwise it would corrupt the frequency characteristic because of the small traffic sample.

A discrete Fourier transformation was used to reveal any dominant frequency
in the signal. The only observable coefficient was equivalent to a one-day period. Other periods were not as dominant or were hidden because of the short measurement interval.

The interesting behaviour of the 24-hour period is that the minimum of daily received spams is located around midnight and the maximum around noon. One can only presume that spammers utilize local zombies that transmit spam when switched on and therefore the characteristics follow the time zone of the victim (GMT+1 in our case) and as a consequence the daily distribution of spam is similar to legitimate emails (y_spam). Further investigation of this behaviour is necessary but is out of the scope of this paper.

4.2 Results of Classification

In this section, we experimented with several configurations of training set to explore characteristics within the context of classification efficiency. The first experiment investigated the fundamental ability of the classifier to distinguish among several classes of SMTP traffic, namely:

1. *ham* – consists of connections that have been successfully received and not
Figure 5. The amount of spam received per hour plotted for the whole measurement period (9 days).

marked as spam by SpamAssassin;
2. \texttt{y\_spam} – connections that have been successfully received and marked as spam by SpamAssassin;
3. \texttt{rejected} – connections that have been rejected because of the DNS black list;
4. \texttt{outgoing} – connections that originated from our server in order to transfer messages to an other email server;
5. \texttt{other} – traffic caused by scanning, DoS, etc.

Despite the fact that connections marked as \texttt{y\_spam} and \texttt{rejected} both constitute spam, it makes sense to assign them into two different classes as it is expected that their connections behave differently and have different fingerprints in network traffic statistics. The distribution of connections into classes is denoted in Table 1.

Table 1. Distribution of annotated SMTP connections into classes.

<table>
<thead>
<tr>
<th>y_spam</th>
<th>ham</th>
<th>rejected</th>
<th>outgoing</th>
<th>other</th>
<th>total amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>11222</td>
<td>1554</td>
<td>38314</td>
<td>2618</td>
<td>4334</td>
<td>58052</td>
</tr>
</tbody>
</table>
In our first experiment, the training set consisted of 66% of all annotated traffic statistics and the remainder served for evaluation of the classifier accuracy. The training of the classification model took approximately 4 minutes while the evaluation was very fast (less than 1 s). Results confirmed that the classification of SMTP traffic into specified categories is possible, moreover quite a small error rate can be achieved (2.1% of incorrectly classified instances). Closer details are presented in Table 2 where each column denotes how many instances of a given category were correctly and incorrectly classified.

Outgoing emails are easily recognized due to the reverse direction of these connections to the remainder. Likewise, the classifier easily recognizes connections that were blocked by DNSBL. On the other hand its ability to distinguish between accepted spams \(y_{spam}\) and legitimate emails \(ham\) is not as bright. If the classifier is trained only to separate these two classes then the results are slightly better but still it would not be wise to detect and block potential spams using this approach. A higher error rate could be caused by similar characteristics of connections, small traffic sample or inaccurate annotation.

Table 2. Confusion matrix of classifier evaluated on 33% of all connections.

<table>
<thead>
<tr>
<th>Classified as</th>
<th>(y_{spam})</th>
<th>ham</th>
<th>rejected</th>
<th>outgoing</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>(y_{spam})</td>
<td>3587</td>
<td>24</td>
<td>76</td>
<td>0</td>
<td>56</td>
</tr>
<tr>
<td>ham</td>
<td>193</td>
<td>540</td>
<td>2</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>rejected</td>
<td>23</td>
<td>0</td>
<td>12949</td>
<td>0</td>
<td>99</td>
</tr>
<tr>
<td>outgoing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>896</td>
<td>0</td>
</tr>
<tr>
<td>other</td>
<td>44</td>
<td>4</td>
<td>135</td>
<td>0</td>
<td>1276</td>
</tr>
<tr>
<td>Precision</td>
<td>93.2%</td>
<td>95.5%</td>
<td>98.4%</td>
<td>100%</td>
<td>88.1%</td>
</tr>
</tbody>
</table>

The goal of our second experiment was to find out whether the classifier would still work if it is trained and evaluated only on the incoming flow of the connection. This experiment was motivated by situations where it is possible to monitor only one direction of the connection, for example due to asymmetric routing. The obtained results are given in Table 3. The performance of the classifier that observes only the incoming stream is nearly the same as for the bidirectional one.

The ageing of the classifier was a point of our interest during the last experiment. The classifier trained on the whole measurement period was evaluated on data (one week) collected two months later. Performance of both classifiers, bidirectional and incoming flow, degraded significantly. The classifier failed to distinguish between spam and legitimate emails. The total classification error was about 6.1%, closer details are given in Table 4.

In order to keep the classifier accuracy high, it should be regularly retrained, we suggest on a weekly basis according to the following experiment. The classifier
was trained on data collected during one day (8th) and evaluated on the rest of the days in sequence. The ability to correctly distinguish between spam and legitimate emails is not significantly influenced during one week (the average error rate is about 7%).

5 Conclusion

This paper presented an alternative approach to spam detection. In comparison to other popular approaches based on content analysis our method utilizes only TCP/IP flow characteristics to reveal spam connections. Measured characteristics were used to train and evaluate the performance of a decision tree classifier. The results of our experiments show that the spam connections are distinguishable from other SMTP communication.

In the near future, we plan to train the classifier on a much larger training set to increase its accuracy. Consequently we intend to use it at the backbone link to classify traffic for more SMTP servers. If it works well we would like to test it as an input for DNSBL techniques to find out the feasibility of such an approach to spam elimination. Other experiments will focus on measurement of spam servers such as the ratio for different target email servers and also on behaviour of spamming hosts, i.e. that spam follows the local timezone of the victim.
References


G3 System – Distributed Measurement Architecture

Tom Košňar

Abstract

The G3 system aims to be a set of complex tools designed for large scale and continuous network infrastructure measurement visualization and reporting. After modifications of internal processing scheme and several months of testing we are able to give a brief description of the G3 system internal architecture that enables measurement in a distributed infrastructure.

Keywords: SNMP, Internet traffic measurement

1 Introduction

We started to introduce the G3 system from its navigation, visualization and reporting capabilities in previous technical reports ([1] and [2]). Now it is time to describe how the system behaves while doing its fundamental task – data measurement, processing and storage. We should go back to the history first and mention our motivation and needs at the beginning of development. Here are our most influential starting points regarding system measurement, processing and storage architecture:

Existing systems experience

We have tested and/or used many infrastructure monitoring systems, based on several concepts, years ago and finished with a home-made system (GT-DMSII - developed and used for many years). We have learnt that we need a system which is really open in terms of measurement – any imaginable method of data collecting must be adopted (although we expected the major role of SNMP – Simple Network Management Protocol). We also wanted to design a system which would give the freedom to redefine (or create for new cases) the structure of collected data and structure of objects/devices that are measured – open tree model (no pairs of values limitations no flat data model nor anything similar).

Open architecture

The system must be open enough to adopt any monitoring requests that can occur in the future and which may lead to measurement of absolutely new groups of information.
Objects identification

We decided to be absolutely independent of any measurement specific identifiers (SNMP indexes for interfaces, hardware components and others for example). The system must keep the possibility to configure the way in which any exclusive identifier is constructed.

Item abstraction

We wanted to have the possibility to merge all items having the same meaning (but different methods of collecting for example) behind a single virtual item when needed.

Dynamic non-aggressive behaviour

There are a lot of items that have to be processed from two consequent steps of data collecting. This is typically the case of SNMP counter type items. The system should be able to catch some dynamics in network infrastructure behaviour while not being too aggressive in terms of requesting data.

Aggregation

Aggregation “by default” was the initial idea – there has to be adequate support for it at the data processing and storage level. For example the system should be able to generate a single aggregated course of temperature regardless of whether it is based on all temperature sensors of a particular device or on a specific sensor of all devices in a computer room.

2 G3 Measurement Architecture

2.1 Processing Framework and Item Definitions

The requirements mentioned above and our reflections led to a specific design. Processing itself is spread between the two basic parts of the system: processing framework and items definitions. The processing framework provides the traditional set of running processes – a real running system consisting of master controlling process, measurement daemons and optional storage daemons – all of them behaving according to the measurement configuration. Item definitions on the other hand are not only configurations of real items to be measured. Items are defined as a hierarchy – they are bound into trees containing three levels of items – group level items, section level items and something like object-id items. Item definitions at any level may contain code or references to library functions (necessary in some cases). While the measurement daemons start to process items in their configured order starting at group level and continuing down in hierarchy,
any item can also run appropriate processing as part of another item regardless of its group or its position in the hierarchy. This freedom enables achievement of absolutely open architecture from a measurement point of view and the only thing that must be kept in mind is timing and the order of actions. On the other hand system complexity is the trade off. The advantage for those who need to extend the system with new groups of information are existing predefined groups of items (System, Interfaces, IP, ICMP, UDP and others) also including vendor specific SNMP MIB subtrees as well as examples of non-SNMP measured items. It is of advantage especially for Perl programmers - whole system as well as item definitions are written in Perl (Practical Extraction and Report Language).

Here is an example of an item tree – an extract of currently defined items including internal item identifier and its mapping to OID (SNMP Object Identifier):

Group: HW
Section: cdspCardStatusEntry
cdspCardLastHiWaterUtilization => 1.3.6.1.4.1.9.9.86.1.1.1.1.4
cdspCardResourceUtilization => 1.3.6.1.4.1.9.9.86.1.1.1.1.3
cdspCardState => 1.3.6.1.4.1.9.9.86.1.1.1.1.2
Section: cerentEnvMonTemperatureStatsEntry
cerentEnvMonTemperatureStatsCurrentValue => 1.3.6.1.4.1.3607.2.80.20.1.30
cerentEnvMonTemperatureStatsDescr => 1.3.6.1.4.1.3607.2.80.20.1.20
cerentEnvMonTemperatureStatsThresholdHigh => 1.3.6.1.4.1.3607.2.80.20.1.40
Section: cesnet_cla_AGC
cesnet_cla_claAGC => 1.3.6.1.4.1.8057.7.20.9.1.2
cesnet_cla_claAGCDescription => 1.3.6.1.4.1.8057.7.20.9.1.3
...

Section: entSensorValueEntry
entSensorPrecision => 1.3.6.1.4.1.9.9.91.1.1.1.1.3
entSensorScale => 1.3.6.1.4.1.9.9.91.1.1.1.1.2
...

Group: ICMP
Section: icmp
icmpInAddrMaskReps => 1.3.6.1.2.1.5.13
icmpInAddrMasks => 1.3.6.1.2.1.5.12
icmpInDestUnreachs => 1.3.6.1.2.1.5.3
...
...
Group: INTERFACE
Section: Dot3StatsEntry
dot3StatsAlignmentErrors => 1.3.6.1.2.1.10.7.2.1.2
dot3StatsCarrierSenseErrors =>
  1.3.6.1.2.1.10.7.2.1.11
dot3StatsDeferredTransmissions =>
  1.3.6.1.2.1.10.7.2.1.7
dot3StatsExcessiveCollisions =>
  1.3.6.1.2.1.10.7.2.1.9
dot3StatsFCSErrors => 1.3.6.1.2.1.10.7.2.1.3
...
...
Section: cMsDwdmFECCurrentEntry
cMsDwdmFECCurrentBitErrCor =>
  1.3.6.1.4.1.3607.2.40.3.2.1.2
cMsDwdmFECCurrentUncorWords =>
  1.3.6.1.4.1.3607.2.40.3.2.1.6
...
...
Section: ifXEntry
ifHCInBroadcastPkts => 1.3.6.1.2.1.31.1.1.1.9
ifHCInMulticastPkts => 1.3.6.1.2.1.31.1.1.1.8
ifHCInOctets => 1.3.6.1.2.1.31.1.1.1.6
...
...
Group: IP
Section: ip
ipForwDatagrams => 1.3.6.1.2.1.4.6
ipFragCreates => 1.3.6.1.2.1.4.19
ipFragFails => 1.3.6.1.2.1.4.18
...
...
Group: SNMP
Section: snmp
2.2 Processing Framework Configuration

The measurement part of the G3 system follows a two-level configuration scheme; master configuration and measured device configuration. The master configuration contains directives for processing the framework and consists of two parts. The common part controls the mechanism of data storage, contains common measurement parameters, location of item definitions, notification directives and other parameters even including functions when needed (see example below). The measurement processes part (‘groups_to_measure’ in example below) controls the number of measurement processes to run, contains the location of appropriate device configurations and also may contain parameters which override relevant settings in the common part. Here is an example of such a master configuration file – its syntax is Perl associative array reference.

```perl
{
    # Item definitions location
    'default_item_dir' => '/data/G3/item_dir',
    'snmp_port' => 161,
    'snmp_max_pdu_len' => 65535,
    'snmp_discovery_step' => 3600,
    'snmp_max_repetitions' => 1024,
    'max_measure_step_time' => 900,
}```
'min_measure_step_time' => 40,
'measure_step_strategy' =>
    ['r', 'r', 'S', 'S', '1', '1', 'r', '1', 'l'],
'filter_clean_unmatching_data' => 0,
'store_data' => 1,
'store_do_fork' => 0,
'max_forked_store_processes' => 1,
'store_root_dir' => '/data/G3/results',
'store_base_directory' => 'boxes',
'max_time_to_store' => 300,
'notification_ttl' => 14400,
'notify_errors_to' => 'somebody@somewhere',
'notify_timeout_to' => 'timeouts@somewhere',
'socket_rrd_storage' => 1,
'socket_rrd_storage_path' =>
    '/data/G3/sock/G3_rrd_storage.sock',
'socket_rrd_storage_read_bufsize' => 8388608,
'socket_rrd_storage_write_bufsize' => 524288,
'socket_rrd_storage_step' => '0.1',
'shmem_rrd_storage' => 0,
'shmem_rrd_storage_max_records' => 8192,
'shmem_rrd_storage_wait_before_send_term_signal' => 30,
'shmem_rrd_storage_max_records_check_step' => 10,
'shmem_rrd_storage_step' => '1.5',
'shmem_rrd_storage_size' => 4096000,
'store_label' => sub {
    package G3_1;
    use strict 'refs';
    my($items, $label, $group) = @_;
    return index(lc $label, 'accounting') >= 0 ?
        ERR() : OK();
}
'groups_to_measure' => [
    {
        'min_measure_step_time' => 300,
        'root_object_dir' => '/data/G3/objects-cfg/0'
    },
]
The measured device configuration file has always the name `.configuration` and is placed in the device directory. This device directory is a dedicated directory which is specified as `root_object_dir` in the master configuration file. The full path of the device configuration file in this example is:

```
data/G3/objects-cfg/6/10.10.1.1/.configuration
```

The device configuration contains the parameters needed to measure the appropriate device and also may contain other parameters overriding inherited parameters from the master configuration. Here is an example of device configuration – its syntax is again Perl associative array reference.

```perl
{
    'snmp_version' => '2c',
    'snmp_community' => 'v2c_some_community',
    'object_description' => 'router-X',
    'snmp_host' => '10.10.1.1',
    'group_topology' => 'Backbone Part NORTH',
    'group_technical' => 'router',
    'store_object_directory' => '10.10.1.1',
    'group_location' => 'PoP in City Y'
}
```

Device configuration may be extended with a filtering configuration. Filtering enables storage of some part of the measured data (specific interface traffic for example) separately under a name dedicated to a specific location. Such a storage location may be shared by the configuration of other devices. It can help for
example to keep long term statistics of external line utilization of multi-homed networks in cases when line end points frequently move from device to device. There is sufficient functionality within the user G3 user interface and G3 reporter to merge and aggregate such courses in any case, but this option makes it independent of erasure of measured data in any participating device. Filter configurations are files in the .FILTER directory which is a sub-directory in the device directory. The filter configuration file name is the root name under which filtered data will be stored. In case we only want to store filtered data we may set the 'filter_clean_unmatching_data' parameter to a non-zero value (globally set in the master configuration above). An example of such a configuration follows – same syntax as before.

```plaintext
{
  # group - base group for which we are currently
  # storing data
  'group' => 'INTERFACE',
  'store_data' => 1,
  # records can be expanded - with key items
  # from 'parent' group
  # ...which is SYSTEM in this case (items like sysName,..)
  'expand_records' => 1,
  # items that should be stored into the data set
  # dedicated to this filter
  'store_label' => sub {
    my ($items,$label,$group)=@_; 
    return OK if ($label eq 'range-of-validity');
    return OK if (index(lc($label), 'keyby')>=0);
    return OK if (index(lc($label), 'bitrate')>=0);
    return OK if (index(lc($label), 'pktrate')>=0);
    return OK if (index(lc($label), 'speed')>=0);
    return OK if ($label eq 'sysName');
    return OK if ($label eq 'sysLocation');
    return ERR;
  },
  # filter matching conditions for one or more instances
  'condition' => sub {
    my $rec=shift;
    # $rec is anonymous hash with measured values
    # we want interface to be UP
    return OK if (
```
value_from_tree($rec,0,['ifAdminStatus'])==1
&&
value_from_tree($rec,0,['ifOperStatus'])==1
&&
(
    # must have one of the following addresses configured
    value_from_tree($rec,',',['ifKeyByIP']) eq '10.10.10.3'
    ||
    value_from_tree($rec,',',['ifKeyByIP']) eq '10.10.11.21'
)
): return ERR;
}
# something like post-processing
'apply' => sub {
    my ($cfg,$data,$items,$filter_group,
    $filter_id,$coordinates)=@_;
    # We require exact count of interfaces that match
    if (scalar @$coordinates != 1) {
        # bad count...
        # we may want to send a message...
        send_filter_match_error_mailmsg($cfg,$data,$items,
            $filter_group,$filter_id,$coordinates);
        # we will remove all marks
        # which were set up in 'condition' function
        remove_filter_signs($cfg,$data,$filter_group,
            $filter_id,$coordinates);
        return OK;
    }
    # We may want to setup administratively interface speed..
    # my ($group,$instance)=@{$coordinates->[0]};
    # my $rec=value_from_tree($data,{},[$group,$instance]);
    # $rec->{IfSpeedAdministrative}=800000000
    #if exists $rec->{IfSpeedAdministrative};
    return OK;
},
# This filter notifications goes there
'notify' => 'errors@somewhere'
2.3 Item Definitions

Item definitions reside in the 'default_item_dir' directory. Each item is defined in a separate file. The syntax of each file is Perl associative array reference. There is no space to describe all options and possibilities here but I’m giving examples of item definitions just to illustrate how the processing is spread among processing framework and items. Here are examples of item hierarchy from the top – 3 items, each representing a different level.

Group level example first:

'SYSTEM' => {
  'section' => 'group',
  'lbl' => 'SYSTEM',
  'group' => 'OBJECT',
  'convert_measured' => sub {
    my ($cfg,$items,$state,$data,$rec,$label)=@_; 
    my $val=value_from_tree($rec,{},[$label]);
    # FIXED instance ID for SYSTEM
    my $instance=(keys %$val)[0];
    # set validity information
    set_value(
      $val->{$instance},
      'range-of-validity',1,
      get_time($val->{$instance},$label,$state)
    );
    return OK;
  },
  'compare-group' => sub {
    use strict;
    use G3_1;
    my ($cfg,$items,$state,$last_data,$data,
         $last_rec,$rec,$label,$key)=@_; 
    my ($instance,$retval,$task);
    foreach $task ('compare-instance','postprocess-instance') {
      $retval=item_compare{
        $cfg,$items,$state,$last_data,$data,
        value_from_tree($last_rec,{},[$key]),
        value_from_tree($rec,{},[$key]),$label,
        (keys %{
          value_from_tree(}
$data,
{}
['SYSTEM'])
})[0],
$task
); return $retval unless $retval==OK;
} return OK;
},'compare-instance' => sub {
  my ($cfg,$items,$state,$last_data,$data,
    $last_rec,$rec,$label,$key)=@_; my $x=value_from_tree($rec,{},[$key]);
  if (scalar keys %$x) {
    # preset -run_counters
    $x->{-run_counters}=1;
  }
  my $all_fields=
    all_exclusive_keys(
      value_from_tree($last_rec,{},[$key]),
      value_from_tree($rec,{},[$key])
    );
  my ($field,$retval);
  foreach $field (@$all_fields) {
    $retval=
      item_compare(
        $cfg,$items,$state,$last_data,$data,
        value_from_tree($last_rec,{},[$key]),
        value_from_tree($rec,{},[$key]),
        master_key($field),
        $field,'prepare-value');
    return $retval unless $retval==OK;
  } return OK;
},'postprocess-instance' => sub {
  my ($cfg,$items,$state,$last_data,
my $all_fields =
    all_exclusive_keys(
        value_from_tree($last_rec,{},[$key]),
        value_from_tree($rec,{},[$key]));
my ($field,$retval);
foreach $field (@$all_fields) {
    next unless exists $items->{$field};
    $retval =
        item_compare($cfg,$items,$state,
            $last_data,$data,
            value_from_tree($last_rec,{},[$key]),
            value_from_tree($rec,{},[$key]),
            master_key($field),
            $field,'postprocess-value');
    return $retval unless $retval==OK;
} return OK;

Section level example second:

'system' => {
    'section' => 'section',
    'lbl' => 'system',
    'group' => 'SYSTEM',
    'index_field' => sub { return 'system' },
    'force_use_iids' => [0],
    'order' => '1',
    'can_create_record' => '1'
}

And like item object-id last:

'sysName' => {
    'section' => 'system',
    'req' => 1,
    'lbl' => 'sysName',
}
2.4 G3 System Generic Architecture

The full scheme of the G3 system is shown in Figure 1. Basic work flow from system start-up is the following:

- the system starts its core – master process and reads the master configuration file (part of the measurement configuration)
- the master process checks and/or sets necessary components (basic directory structure, log mechanism, validity of item definitions path and others)
- the master process starts the appropriate common storage process if needed (in case of non-zero socket_rrd_storage or shmem_rrd_storage parameter)
the master process parses `groups_to_measure` section in the master configuration and starts the child measurement process – `measurement module` for each record found

- the measurement process (`measurement module`) checks (periodically) the appropriate `root_object_dir` for device configuration files, last measurement results files, and measurement state files (parts of `measurement configuration`)
- the measurement process (`measurement module`) serializes – after measuring each device – device configurations according to the requested measurement start times and starts to measure the first device in the queue
- collected and processed data are stored (using a configured mechanism), the device state file and the last measurement result file are updated and the device configuration is inserted into the queue according to the next measurement time (fixed or generated – depending on `measure_step_strategy` configuration parameter)

There are several mechanisms how the measurement process stores collected and processed data – everything depends on a combination of basic storage parameters. The following list describes system behaviour when the appropriate parameters are set (non-zero value means ON):

`store_data`
setting this parameter is a necessity otherwise NO data will be stored; switching it off may make sense for measurement tuning or for operational monitoring when item definitions may generate alarms or notifications but without any need to store data

`store_do_fork`
the measurement process forks immediately after collecting and processing data (regardless of storage method used); the child process starts to store the data while the parent process immediately starts to measure the next device in the queue (it may be the same device again); this method may be used when a very short time step is requested (very low values of `min_measure_step_time`)

`shmem_rrd_storage`
the measurement process (or its child optionally) writes processed data to a shared memory segment and starts to measure the next device in the queue (or exits when being a child); the storage process (`storage`) periodically checks the shared memory segment, writes data found to a real storage area and cleans the memory

`socket_rrd_storage`
the measurement process (or its child optionally) writes processed data to the
UNIX socket and starts to measure the next device in the queue (or exits when being a child); storage process (storage) periodically checks the other side of the socket and writes received data to real storage.

nothing set

the measurement process (or its child optionally) stores processed data itself; in this case storing data without forking may delay measurement of the next device in the queue.

Probably the best setup for large (> $10^5$ items) or “aggressive” (frequent measurement) configurations is socket_rrd_storage without store_do_fork. Choosing optimal parameters also depends on average device response times. In general the essential parameter is the proportion between the number of parallel measurements (the number of measurement processes is unlimited) and the corresponding amount of data to be stored on one side and throughput of real storage mechanism including I/O at OS level, databases, data formats, supporting tool efficiency and others. G3 system uses two types of databases. The first one is proprietary (Perl Storable based) and is used for text items. It is fast and represents small volumes of data only – no break in the storage process. The second one is RRD (Round Robin Database) based on rrdtool packages and RRDs Perl API. The G3 system is configured to store multiple items (currently 16) into a single RRD to speed up things. Although the RRA (Round Robin Archive) structure may be set up in configuration we usually use the system default setup. It (single RRA) currently consists of 7 aggregation intervals with approximately 3400 rows (summary) and three values per row (minimum, maximum, average) covering a 12 year period. There also has to be adequate support (especially for large configurations) at the operating system level to use resources effectively. There are parameters that depend on local conditions and monitoring purposes, but there are common rules that should be kept in mind anytime when setting up monitoring with RRD based storage – I can absolutely recommend reading papers like [3] – and provide step-by-step testing when preparing critical RRD based installations. Our production single host installation currently runs on 4xQuad Core Xeon X7350, 64GB RAM, 8xSAS 15K 73GB drive in RAID10 (64K chunk, adaptive read-ahead, write-back cache at the controller) with Debian-etch installed. The OS scheduler is set to noop, file-system read-ahead is set to 16KB, the number of requests in the queue is set to 512. This installation runs the G3 system currently with 14 parallel measurement processes running. Data are stored through UNIX socket mechanism. The overall number of measured devices is currently around 120 and total number of items measured is around 550000. The average measurement time step is 750 seconds with current limits at 40 and 1200 seconds as you can see in Figure 2.
2.5 Distributed G3 System Measurement

There are cases when single host installation is not suitable or not efficient enough for what we need. Reasons may be for example - not powerful enough hardware available, absolutely different monitoring requirements on devices groups to be measured, requested aggressive and/or separate measurement of some device and others. There is always a chance to install multiple instances of monitoring system but on the other hand users probably do not prefer a solution with several monitoring hosts, each with its own user interface – its confusing. Therefore we improved the G3 system to satisfy requirements on separate measurements while keeping a single user interface that merges data of all devices regardless of where they were measured. The only limitation was the necessity of at least read-only access to all measured data at file-system level for user interface (G3 UI module) – otherwise we would lose native functionality of rrdtool packages especially when running aggregated data retrieval (which is something we definitely need). The basic scheme of the G3 system in distributed measurement architecture is shown in Figure 3 – it is just an extension of the generic scheme shown in Figure 2.

There are no changes visible in Figure 3; only add-ons. Orange lines show read-only NFS mapping of a part of the measuring host’s storage to the host where the user interface runs. There were a lot of changes made, but they are hidden deeply in the G3 system libraries. The measured data contain the full path from the G3 storage root at the measuring host – there are many purposes for it – and we had to develop automated path re-mapping mechanisms that translates path derived identifiers on-the-fly and only when needed. Adequate extensions were done in functions preparing source data for navigation tree generation and other extensions in measurement configuration to give enough flexibility in measurement setup. The current state of the G3 system enables running of distributed
measurement in both basic modes - separate measurement of groups of devices and separate measurement of device components.

2.5.1 Separate Measurement of Groups of Devices

This distributed measurement mode is the simplest one. Measurement configuration may be in principle the same as in single host measurement – no extensions are needed. A device distribution scheme as well as data merging is shown in Figure 4.

There are only two things that must be done to make the whole system functional in this configuration:
— the appropriate part (results of measurement) of measuring only host storage must be mapped to the proper place (the directory where the tree with results
begins) under any name at the host running user interface
— a mapped directory must be added to the configuration of the user interface
(to be used when constructing a navigation tree or when accessing stored data)

File-system mapping may look with respect to Figure 3 and configuration examples above like this:
Y:/data/G3/results/boxes -> X:/data/G3/results/boxes.Y

And configuration file for interface modules at host X may look like this:
{
dirs => {
  # results root directory
  '/data/G3/results/boxes.Y' => {
    # shall be used ?
    active => 1
  },
  '/data/G3/results/boxes' => {
  },
}
2.5.2 Separate Measurement of Device Components

This distribution is more interesting from the configuration point of view. The distribution scheme is shown in Figure 5.

Figure 5. G3 system distributed measurement – groups of device components.

In the above example, we may want to measure *Interfaces* and *Streams* (correspond with *INTERFACE* and *VSTREAM* classes of objects) from one host and the rest from another host. We must configure both measurements carefully to avoid duplications. For example multiple measurement of items that are stored into *RRD* databases may cause multiplication of values at the user interface level.
because the system provides automated aggregations. Therefore we must explicitly configure which items to measure at each host – in the master configuration. The important part of the master measurement configuration file for Host X may look like this:

```
{
    ...
    ...
    # - sign (standalone) switches everything off
    # + sign with item identifier enables items
    # (enabling group level items enables all their children)
    'wanted_items_condition' => '-,+INTERFACE,+VSTREAM',
    ...
    ...
    'store_label' => sub {
        package G3_1;
        use strict 'refs';
        my($items, $label, $group) = @_;  
        # some special items are always on
        # (cannot be switched off)
        # we may choose which of them to store
        return OK() if index($label, 'sysMeasureStep') >= 0;
        return OK() if index($label, 'sysMeasureTimeSpent') >= 0;
        return OK() if index($label, 'sysMeasureDelay') >= 0;
        # we don’t want to run exact interface accounting
        return ERR() if index(lc $label, 'accounting') >= 0;
        # finally we disable everything
        # except 'INTERFACE' and 'VSTREAM' groups
        return ERR() if $group ne 'INTERFACE'
        &&
            $group ne 'VSTREAM';
        # OK for 'INTERFACE' and 'VSTREAM' groups
        return OK();
    },
    ...
    ...
}
```
Configuration for Host Y is complementary and will look like this:
G3 System – Distributed Measurement Architecture

```perl
{
...
...

'wanted_items_condition' =>
  '-.,+HW,+SYSTEM,+UDP,+ICMP,+SNMP,+IP',
'store_label' => sub {
  package G3_1;
  use strict 'refs';
  my($items, $label, $group) = @_;  
  return ERR() if $group eq 'INTERFACE' 
      || $group eq 'VSTREAM';
  return OK();
},
...
...

File-system mapping as well as configuration of interface modules will be the same as in the previous case.

We tested this setup (separate measurement of device components) for three months (after stabilizing the extended system). We tested not only interactive user interface functionality but also G3 reporter in this configuration. Experimental maps of network utilization and network health were periodically generated from this distributed configuration during that period.

3 Conclusion

The G3 system is used for large scale infrastructure monitoring of the CESNET2 network. As the backbone network becomes more hybrid containing L1–L3 devices we observe growth of items that are requested to be measured. Especially the number of optical parameters of DWDM systems (including our locally developed optical amplifiers) or the number of monitored sensors is evidently growing. We have to prepare scenarios for when a not powerful enough single hardware will be available (with an acceptable price-to-performance ratio) to measure all required items from the whole set of devices. The distributed architecture of the infrastructure monitoring system is a step towards a solution.
References


[3] Plonka, D.; Gupta, A.; Carder, D. Application buffer-cache management for performance: running the world’s largest MRTG. In 21st Large Installation System Administration Conference (LISA ’07), Dallas (TX), 11-16 November 2007. Available online\textsuperscript{3}.

\textsuperscript{1} http://www.cesnet.cz/doc/techzpravy/2005/g3/
\textsuperscript{2} http://www.cesnet.cz/doc/techzpravy/2007/g3-reporter/
\textsuperscript{3} http://www.usenix.org/events/lisa07/tech/full_papers/plonka/plonka.pdf
Designing a Hardware-Accelerated Firewall with Two 10 Gbps Ports

Víktor Puš, Tomáš Dedek

Abstract

High-speed packet filtering should be one of the first steps in securing any modern computer network. However, solutions over 1 Gbps are practically impossible to implement in software, and must be implemented with the use of specialized hardware. This paper describes the design of a two-port firewall for 10 Gbps networks. This solution is based on hardware implementation of our classification algorithm. The firewall is designed to process data at full speed, without any packet loss. The target platform is the COMBOv2 card.

Keywords: FPGA, hardware acceleration, firewall

1 Introduction

With the increasing speeds of computer networks, tasks with extremely high demands on computational power appeared. Examples of such tasks are pattern matching, packet classification, or even packet routing. However, implementation of these tasks as a software application for common processor could be very problematic or even impossible, due to a very high packet rate. On the 10 Gbps network, the packet rate is up to 15 million packets per second, so one packet is processed every 67 ns. Because speed is impossible to achieve with general CPUs, hardware implementations of these tasks are required. For research purposes, FPGA chips are a good choice, because of the possibility of reconfiguring the device many times.

1.1 Packet Classification

We focused on one of these performance-demanding tasks: packet classification. The most common application for packet classification is packet filtering. Packet filters are used not only as firewalls protecting a network or computer. Another useful application is the lawful intercept, where network traffic is monitored and selected packets are sent to be further processed or stored. The classification algorithm is based on the following steps:

— The algorithm stores a set of rules ordered by priority.
The resulting rule number is the first rule number matching the packet. Each rule has a defined action, which instructs the firewall about what to do with the packet. The basic actions are Accept or Deny for traffic filtering. Nevertheless, actions could also be more complex.

After the packet is classified and the requested action is performed, the packet is sent to one or more output interfaces. The selection of output interfaces could be also a part of the action.

All these steps must be performed in just nanoseconds, so a parallel hardware implementation is appropriate. Many papers about packet classification have been published, a good overview of the current state is [7], with some recent improvements in [2] and [3]. Classification solutions may be divided into two groups:

- Technical solutions, using Ternary Content-Associative Memories (TCAMs).
- Algorithmic solutions without TCAMs.

TCAM solutions have one great advantage: the constant processing time. The property of Content-Addressable Memory is that it compares the input word to the whole content of the memory in a single access. This way, packets are classified at a constant speed. However, TCAMs have many drawbacks, namely the price and high power consumption. That’s why algorithmic solutions are a research subject. Algorithmic solutions use cheaper Random Access Memory, but their performance and memory requirements are often hard to predict. Our algorithm reaches the constant time complexity, so that it can compete with TCAM solutions.

### 1.2 Firewall

Current firewalls may be divided into three main groups:

- **Stateless firewalls** process every packet independent of the previous packets.
- **Stateful firewalls** store the state of every connection and process traffic based on the state of the connection.
- **Application firewalls** analyze application protocols and do application-specific filtering. Example: Email filter.

We chose to implement the stateless firewall, because of its straightforward design and the clear requirements for its function.

Every stateless firewall has to perform several actions. When the packet is received, its header is parsed and important header fields are extracted. The packet is then classified on the basis of extracted header information. The requested action is known after classification, so that the firewall must perform the action (drop or forward packet, or other tasks).
Our goal is to design a hardware accelerated firewall with two 10 Gbps ports on the PC platform. We intend to use COMBOv2 acceleration cards [1] for evaluation. The classification algorithm is implemented in the Virtex5 FPGA and necessary data structures are stored in the external QDR-II SRAM memory.

The rest of this report is organized as follows: in the next section we describe the overall system architecture and give examples of its use. Section 3 describes our classification algorithm in detail. The firmware architecture is proposed in the section 4. Section 5 summarizes our work and Section 6 concludes the paper.

2 System Architecture

The firewall consists of several layers (see Figure 1). The hardware layer is the COMBOv2 card [1]. It is responsible for the physical and electrical environment. The firmware layer is the configuration of the Virtex5 FPGA. This is where the accelerated part of the firewall is placed. Packets are received and classified in the Firmware layer. Also the requested action is performed directly in the FPGA. The Kernel layer is a Linux kernel module, which is responsible for firmware configuration and communication with the user space. And finally, the user space layer is a set of software tools and libraries for system management and possibly packet processing.

![Figure 1. Layer structure of the firewall](image)

We propose two basic scenarios for the firewall configuration:
2.1 Local Configuration

In this simpler option, the firewall is configured from a configuration file containing the ruleset. A set of command-line tools is ready for firmware initialization, ruleset loading, statistics gathering etc. Figure 2 shows the nificgend daemon, which prepares all necessary data structures and configures the classification core of the firewall. The second tool in Figure 2 is the nific-config, which is used by the user to load a ruleset into the nificgend.

```
BPF config file
  ▼
 nific-config
  ▼
nificgend
  ▼
COMBO card
```

**Figure 2.** Local configuration of the firewall

2.2 Remote Configuration

This option may be useful for automated and remote control of the firewall. It is especially well suited for legal intercept applications. Figure 3 shows the system architecture. The nificgend daemon is also used here, but it receives its configuration data from the remote Control Centre over the NETCONF protocol [8]. The nificexp daemons are responsible for forwarding selected packets to the remote Monitoring Centre. The nificd daemon is a mediator of configuration data among all parts of the system.

2.3 Examples of Use

The proposed system may be used in several scenarios, we will discuss three of them here. The most simple is the firewall, filtering traffic on the edge of the
network (Figure 4). Rules are written in a format similar to BSD Packet Filter, which is widely accepted in the industry. Another option is a passive network probe, see Figure 5. In this case, network interfaces are used only to receive mirrored data from any network. The filtering
functionality is used to select only relevant packets. Unimportant packets are dropped in the Firmware layer, so that host system (user space layer) receives only a fraction of the network traffic for further software analysis or storage.

**Figure 5.** Device configured as a network traffic analyzing probe

The third possibility is the legal interception scenario, where the system forwards filtered packets to the remote Monitoring Centres for further analysis or storage, see Figure 6.

**Figure 6.** Device with remote configuration and monitoring centres
3 Packet Classification Algorithm

Our algorithm is based on problem decomposition, similar to several previously published approaches [2] and [3]. The algorithm scheme is shown in Figure 7.

The first step is the parallel processing of all significant packet header fields by the LPM algorithm. From a given set of prefixes with various lengths, the LPM algorithm selects the one that best matches the given full-length value. Many advanced LPM algorithms are based on trie, but use additional improvements. We use the TreeBitmap algorithm [5] to perform the Longest Prefix Match operation for IP and TCP/UDP ports. For this reason, TCP/UDP port ranges are converted to prefixes (one range is converted to one or more prefixes). Other field are processed in simple modules like tables for protocol number and input interface number processing, and CAMs for MAC address processing.

![Figure 7. Scheme of the classification algorithm](image)

All results of the first processing stage are concatenated into one 67 bit-wide word. Detailed analysis shows that several words may correspond to one rule. These words are called pseudo-rules in the literature [2]. Figure 8 illustrates how pseudo-rules are generated.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dimension 1</th>
<th>Dimension 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>1*</td>
<td>*</td>
</tr>
<tr>
<td>R2</td>
<td>1*</td>
<td>00*</td>
</tr>
<tr>
<td>R3</td>
<td>101*</td>
<td>100*</td>
</tr>
</tbody>
</table>
We can see attempts at classification in two dimensions, with three rules R1, R2, R3 (Table 1). But some combinations of LPM results are not covered, although they must also produce the correct rule number. For example, the combination of LPM results; 101 in Dimension 1 and 00 in Dimension 2, is not in the Rule Table, but the correct result is the rule R2. That is why three pseudo-rules P1, P2, P3 must be added to accommodate the uncovered combinations. Table 2 contains these pseudo-rules.

**Table 2. Three added pseudo-rules**

<table>
<thead>
<tr>
<th>Pseudo-rule</th>
<th>Dimension 1</th>
<th>Dimension 2</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1*</td>
<td>100*</td>
<td>R1</td>
</tr>
<tr>
<td>P2</td>
<td>101*</td>
<td>00*</td>
<td>R2</td>
</tr>
<tr>
<td>P3</td>
<td>101*</td>
<td>*</td>
<td>R1</td>
</tr>
</tbody>
</table>

We constructed a special hash function using the algorithm for the perfect hash function [9] which uses the random acyclic graph search method. This hash function has intended collisions, all pseudo-rules corresponding to one rule are hashed to the same output number. In our example, LPM results (1*, 00*) and (101*, 00*) are both hashed to the address of rule R2. This way, the algorithm performs rule lookup in constant time and does not need to store any pseudo-rules. To compute the constructed hash function, two memory accesses must be performed for every input word. The size of the perfect hash function data structure depends on the amount of pseudo-rules. Because it could be very large, an external memory is used.

As the hash function gives some result for each packet – even if a packet
Designing a Hardware-Accelerated Firewall with Two 10 Gbps Ports 109

matches no rule – the false positive error could occur. This issue is resolved in the third step of the algorithm, where the packet header is compared to the selected rule. If the packet header fits the rule, the correct rule was found. If not, the packet does not match any rule.

An important feature of our approach is massive software preprocessing that has to be done for every new firewall configuration. The preprocessing includes content generation for all packet header processing components, the hash function memory, the rule table and much more. The main idea is to do as much work as possible in advance (in software) to save scarce FPGA resources.

4 Firmware Architecture

The NetCOPE platform [4], which includes Input/Output Buffers and DMA controllers, is taken as a basic platform for our design, so we have a certain level of abstraction upon interface. A hardware scheme of the proposed solution is in Figure 9. The whole figure shows the packet processing pipeline. As can be observed, the system has internally three pathways, instead of two. This is because the PCIe connection – hardware acceleration card is plugged into the PC, and software applications can also receive and send packets. All three lines are internally implemented by the FrameLink protocol in the FPGA. The FrameLink protocol is a simple modification of the Xilinx LocalLink protocol\(^1\). Each FrameLink frame encapsulates one Ethernet frame and adds 128 bits of FrameLink header and a few service signals. To transmit full 10 Gbps flow, each line is 128 data bits wide and the whole design is running at a frequency of 125 MHz.

Packets from the network are received by XGMII IBUF modules and converted into the FrameLink format. Packets sent from software are received in the DMA TX Buffer (part of the NetCOPE platform, [4]) and also converted into FrameLink. Three FrameLink flows then continue to three instances of the Header Field Extractor engines, which perform packet header parsing. Further, each flow is forked into two parts: The first is the original packet encapsulated in the FrameLink protocol. The second is also FrameLink protocol, but it contains only parsed header fields.

The original packets are stored in three on-chip FIFO queues, implemented by Virtex5 BlockRAMs.Parsed packet header data are sent to the Classification module. The module is a straightforward implementation of the algorithm described in the previous section. The output of the Classification module are the result of the classification algorithm: the number of the correct matching rule.

\(^1\) http://www.xilinx.com/products/ipcenter/LocalLink_UserInterface.htm
and its associated actions. The Header Insert module inserts a rule number and actions into the FrameLink header.

According to the action in the frame header, the Crossbar module switches each frame from the input line to the correct output lines. Each frame can be switched to zero (the packet is dropped) or more interfaces. Even if the Crossbar has buffers in each intersection (nine buffers), the output blocking could occur when two or three input lines are switched into one output line. If this happens, the Deficit Round Robin algorithm [10] is used to perform fair queueing in the Crossbar. The output branch to software, has an additional module, the Trimming Unit. This module trims packets according to action. This may be useful for saving the PCIe throughput. The flow to software is then stored in the DMA RX Buffer, from where it is then transmitted to the software processing. Two other flows use XGMII OBUF to send packets into the network.

**Figure 9.** Firewall architecture
5 Results

The whole classification core, including everything between Header Field Extractor and output interfaces, was verified by a modified SystemVerilog verification environment as described in [6]. The input to the verification environment is the rule set. On the basis of the rule set, the set of testing packet headers is generated and is used as an input to the design. Generated packet headers are then independently classified by the verification environment and the results are compared to the results from the design. This approach helped us to find a lot of tricky bugs before the design was tested in hardware. The SystemVerilog verification significantly decreased the development time of the whole system.

Our implementation classifies packets according to these nine header fields:

- Source MAC address
- Destination MAC address
- Source IPv4 address
- Destination IPv4 address
- Protocol
- Source Port
- Destination Port
- TCP Flags
- Input Interface number

Our tests with synthetic and real-life rulesets from a university campus network show that the device is able to support up to 1000 rules. There are several factors limiting the number of rules, the most severe limitation is the perfect hash table. Even when it is stored in external QDR-II memory, the number of pseudo-rules may be too high.

We have tested a prototype of our design with 1 Gbps only, because the 10 Gbps NetCOPE platform on the COMBOv2 card is still under development, so we don’t have results regarding throughput of the system. However, the number of FPGA resources is known (at least approximately), because 10 Gbps and 1 Gbps versions are very similar. The whole design in the Virtex5 LX110T FPGA consumes 16,872 (97%) slices and 126 (85%) BlockRAMs.

6 Conclusion

Our results show that programmable hardware is able to classify packets even in 10-Gbps networks by exploiting the hardware parallelism (parallel execution units, pipelined processing). Our implementation on the COMBOv2 card was verified in SystemVerilog and passed basic functionality tests.

We continue to improve the classification algorithm by reducing the number
of pseudo-rules and optimizing the implementation to support more rules with the same hardware. An example of these optimizations may be Rule Table compression or a more memory-efficient LPM algorithm than TreeBitmap. We also plan to support the IPv6 protocol.

References


⁵ http://www.ietf.org/rfc/rfc4741.txt
Impact of Network Security on Speech Quality

MIROSĽAV VOZŇÁK

Abstract
This technical report deals with the impact of a secured network environment on speech quality. Here-in are presented the results of analyzing voice over secure communication links based on TLS, especially in an OpenVPN solution. The use of secure network environments can affect speech quality. Included in this report is the performance comparison of cipher algorithms and a description how the security mechanisms used influence the final R-factor. The presented results are based on experiments which have been performed in a real IP network.

Keywords: TLS, OpenVPN, E-model

1 Introduction
Security is becoming a necessity of current corporate networks and two solutions appear, either based on IPsec or TLS, and therefore this topic has been chosen for our research in the area of speech quality. Research regarding Voice over IPsec has been performed at the University of Milan [1], so I decided to focus on TLS. Together with my colleagues from Milan we established our real platform, in order to understand how the voice services in an IP network are affected by using a secure IP environment. The real performance test was implemented between VŠB-Technical University of Ostrava and Universita degli Studi of Milan. We pointed out the advantages and the disadvantages of the adopted security measure. We described two virtual testbeds, one developed using a traffic emulator and the second one based on a network simulator. Both virtual environments were implemented in secure and insecure processes. The executed measurements proved the obvious impact of some secure solutions on voice quality, the results have been published [2] but without an exact calculation. A method of calculation describing the overall impact of security on speech quality is published in this technical report, the method is valid for TLS and has been tested with OpenVPN.

Virtual Private Network (VPN) is a technology for constructing a private network over public networks. OpenVPN is one of the most popular software-based VPN products and has high flexibility. The usability of OpenVPN is high because it offers an open-source, cost-effective and widely tested solution, not requiring expert knowledge. Software VPN products are popular, because they don't need any appliance and OpenVPN provides such a solution which is based on matured protocols. The OpenVPN security model is based on SSL (Secure Socket Layer),
the industry standard for secure communications via the IP network. OpenVPN implements a transport secure network extension using the TLS protocol (Transport Layer Security).

On the other hand the usage of OpenVPN increases overhead which is affected by encryption and this overhead can influence overall speech quality [3]. This paper contains a description of OpenVPN and its possibilities regarding configuration, then there is explained the core of the matter; the splitting of an RTP packet into equally divided blocks.

2 OpenVPN and Encryption

TLS ensures a secured connection which is encrypted and decrypted with the keys negotiated during a phase of key exchange. The key exchange and authentication algorithms are typically public key algorithms but subsequent data exchange is usually done by symmetric ciphers because of considerably faster processing. Of course, symmetric encryption is more suitable for IP telephony and this paper deals only with this type of cipher [4]. TLS involves three main phases such as negotiation of supported algorithms, key exchange and authentication, and finally symmetric encryption of transmitted data.

The endpoints establishing the VPN tunnel are declared one as the server and the other as client. Before establishing the VPN, the client first reaches the server on a specific port, whereas the server doesn't need to reach the client. Configuration files are located in the directory `/etc/openvpn` as `server.conf` or `client.conf`. The tunnel can be established on UDP or TCP, unfortunately the TCP protocol is more widespread although UDP is more effective because of real-time applications. The most important information in configuration files is the type of cipher algorithm because it affects the number of blocks and overhead as is shown in Figure 1, which illustrates the splitting of one RTP packet to N blocks. Every block has the same length which contains in the case of AES (Advanced Encryption Standard) 128 bits, although the key size can be not only 128 bits, but also 192 or 256 bits. If other algorithms are applied such as DES (Data Encryption Standard), Triple DES or BF (Blowfish), then the block size is set to a value of 64 bits. A complete list of supported cipher algorithms can be obtained as a result of the command `openvpn --show-ciphers`. The following ciphers and cipher modes are available for use with OpenVPN. Each cipher shown below may be used as a parameter to the `--cipher` option:

- DES-CBC 64 bit default key (fixed)
- RC2-CBC 128 bit default key (variable)
- DES-EDE-CBC 128 bit default key (fixed)
Figure 1. The number of blocks affected by CBC mode.

- DES-EDE3-CBC 192 bit default key (fixed)
- DESX-CBC 192 bit default key (fixed)
- BF-CBC 128 bit default key (variable)
- RC2-40-CBC 40 bit default key (variable)
- CAST5-CBC 128 bit default key (variable)
- RC2-64-CBC 64 bit default key (variable)
- AES-128-CBC 128 bit default key (fixed)
- AES-192-CBC 192 bit default key (fixed)
- AES-256-CBC 256 bit default key (fixed)

The default key size is shown as well as it can be changed with the --keysize directive; using the CBC mode is recommended. CBC means Cipher-block chaining. In this mode of operation, each block of plain text is XORed with the previous cipher text block and afterwards is encrypted. That is why an initialization vector IV must be used in the first block, see Figure 1.
3 Used Techniques of Measurement

The results presented here are based on a series of measurements which have been performed in a real network with OpenVPN and IxChariot[^1], a scheme is shown in Figure 2.

![Logical scheme of testbed.](image)

The whole traffic carried out between an OpenVPN client and server was captured by Wireshark and individual packets were analyzed. IxChariot is a software, produced by Ixia, which consists of the IxChariot console and IxChariot endpoints. The IxChariot console allows a selection of several test configurations, such as the codec used, timing, number of concurrent calls, test duration and so on. The test is initialized at the console, the conditions are uploaded into endpoints and consequently the test is performed. The results are sent back to the console. There was observed an influence of OpenVPN-TLS on overhead which has been increased and hence the required bandwidth has been affected.

4 Bandwidth Requirements

The basic steps of speech processing on the transmission side are encoding and packetizing [5], [6]. RTP packets are sent in dedicated times and the difference between them depends on timing. This process of packetizing is given by the following basic equation:

$$\Delta t = \frac{P_S}{C_K}$$

where $\Delta t$ is timing in seconds, $P_s$ is payload size and $C_R$ represents codec rate. The timing can be derived from the content of the RTP packet as the difference of two consecutive timestamps, see Equation 2. The typical value of the sampling frequency is 8 KHz.

$$\Delta t = \frac{\text{timestamp}_{(N+1)} - \text{timestamp}_{(N+1)}}{\text{sampling\_frequency}}$$

It is necessary to express the packet size at the application layer which might be defined by the following formula:

$$S_{AL} = H_{RTP} + P_s$$

where $S_{AL}$ is the expected size that consists of an RTP header $H_{RTP}$ and payload size $P_s$. Equation 4 determines the size $S_F$ of the link layer frame.

$$S_F = S_{AL} + \sum_{j=1}^{3} H_j$$

$S_F$ includes a packet at the application layer and the sum of lower located headers of the OSI model where $H_1$ is the media access layer header, $H_2$ the Internet layer header and $H_3$ is the transport layer header.

Figure 3. Bandwidth as a function of payload size and concurrent calls.
Figure 3 illustrates the relationship between bandwidth, payload size and number of concurrent calls.

\[
BW_M = \sum_{i=1}^{M} \frac{S_{ri}}{\Delta t_i}
\]  \hspace{1cm} (5)

In Equation 5 we express total bandwidth \(BW_M\) [kbps] required in the case of \(M\) concurrent calls. If we now apply Equations 1, 2 and 4 to Equation 5, we obtain the following result:

\[
BW_M = MC_R \left( 1 + \frac{H_{\text{esp}} + \sum_{j=1}^{3} H_{j}}{P_S} \right)
\]  \hspace{1cm} (6)

We have to realize that TLS is located between two layers of the OSI model, between application and transport layer and therefore we apply \(S_{\text{TLS}}\) instead of \(S_{\text{AL}}\). This replacement should be done with respect to the denoted location of TLS and we define a new parameter \(S_{\text{TLS}}\), size at TLS layer. \(S_{\text{TLS}}\) is expressed in Equation 7.

\[
S_{\text{TLS}} = C_0 + \left\lceil \frac{S_{\text{AL}}}{B_S} \right\rceil 
\times B_S
\]  \hspace{1cm} (7)

where we used the ceiling function which gives the smallest integer greater than or equal to its argument:

\[
\lceil x \rceil = \min\{n \in \mathbb{Z} | x \leq n\}
\]  \hspace{1cm} (8)

The ceiling function was defined by M. Schroeder in 1991 [7] and the symbol was coined by K. Iverson in 1994. The parameter \(B_S\) represents a block size which has been explained in Figure 1, its value is 64 or 128 bits and depends on the applied cipher algorithm (AES, DES, Triple DES or Blowfish). \(C_0\) is a constant and equals to zero in the case of clear TLS. Unfortunately OpenVPN adds supplementary overhead that is included in \(C_0\). The value has been achieved by performing experiments, see Figure 2. We can claim that this constant \(C_0\) is 83 bytes in the case of a block size of 128 bits and 75 bytes in the case of block size of 64 bits.

5 Achieved Results

Relations stated in the previous chapter have been confirmed by experimentation. Figures 4 and 5 illustrate how required bandwidth is affected by TLS and OpenVPN.
The first column of Table 1 contains the codec G.729 and both variants of G.723.1. The block size has a length of either 64 or 128 bits. Table 1 provides the results for Ethernet without TLS, with TLS and with OpenVPN [8], [9].

6 Impact on R-factor

Lack of bandwidth causes a loss in the first case, hence the estimation of its impact on R-factor is explained in this chapter. The maximum value of R-factor for narrowband codecs is 94, the overall quality (R-factor) is calculated by estimating the signal-to-noise ratio of a connection ($R_0$), subtracting the network impairments ($I_S$, $I_D$, $I_{IE-EF}$), and adding an Advantage factor $A$.

$$R = R_0 - I_S - I_D - I_{IE-EF} + A$$  \hspace{1cm} (9)

The first item; $R_0$ is derived from the original SNR, the second; $I_S$ considers non-optimum sidetone, quantizing distortion, overall loudness and other impairments which occur more or less simultaneously with the voice transmission. The delay impairments are included in the parameter $I_D$ as a mathematical summary.
of transmission delay, talker echo and sidetone. The effective equipment $I_{E-EF}$ is an equipment impairment that considers the influence of used codecs and impairments due to packet loss and rejection. The packet loss distribution can be modelled using a Markov process. A multi-state Markov Model is used to measure the distribution of lost or discarded packets or frames, and to divide the call into “bursts” and “gaps”. The call quality is calculated separately in each state and then combined using a perceptual model, such as in VQmon [10]. The VQmon mentioned incorporates a G.107 compliant implementation of the E-Model. However, I applied a very simple method described in the last revision of G.107 from 2005 [11]. The impairment factor values $I_{E}$ under packet-loss were tabulated for particular codecs. Robustness Factor; $B_{pl}$, is defined as a codec-specific value and can be described as the robustness of the codec to packet-loss. Both values are listed in Appendix I of ITU-T G.113 and are available for several codecs. If we consider the Packet-loss Probability as $P_{pl}$, the $I_{E-EF}$ impairment factor can be calculated using the formula:

**Figure 5.** Comparison of bandwidth for codec G.711 without TLS, with TLS and OpenVPN, $B_S = 128$ bits
Impact of Network Security on Speech Quality

Table 1. Values of required bandwidth for various environments

<table>
<thead>
<tr>
<th>codec</th>
<th>block size</th>
<th>timing</th>
<th>no TLS</th>
<th>TLS</th>
<th>OpenVPN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[bits]</td>
<td>[ms]</td>
<td>[kbps]</td>
<td>[kbps]</td>
<td>[kbps]</td>
</tr>
<tr>
<td>G.723.1/6.3</td>
<td>128</td>
<td>30</td>
<td>24</td>
<td>30.4</td>
<td>52.53</td>
</tr>
<tr>
<td>G.723.1/6.3</td>
<td>128</td>
<td>60</td>
<td>15.2</td>
<td>17.33</td>
<td>28.4</td>
</tr>
<tr>
<td>G.723.1/5.3</td>
<td>128</td>
<td>30</td>
<td>22.93</td>
<td>26.13</td>
<td>48.27</td>
</tr>
<tr>
<td>G.723.1/6.3</td>
<td>64</td>
<td>60</td>
<td>14.13</td>
<td>17.33</td>
<td>28.4</td>
</tr>
<tr>
<td>G.723.1/6.3</td>
<td>64</td>
<td>30</td>
<td>24</td>
<td>28.27</td>
<td>50.4</td>
</tr>
<tr>
<td>G.723.1/5.3</td>
<td>128</td>
<td>60</td>
<td>15.2</td>
<td>17.33</td>
<td>28.4</td>
</tr>
<tr>
<td>G.723.1/6.3</td>
<td>64</td>
<td>60</td>
<td>22.93</td>
<td>26.13</td>
<td>48.27</td>
</tr>
<tr>
<td>G.723.1/6.3</td>
<td>64</td>
<td>30</td>
<td>14.13</td>
<td>16.27</td>
<td>27.33</td>
</tr>
<tr>
<td>G.729</td>
<td>128</td>
<td>10</td>
<td>60.8</td>
<td>78.4</td>
<td>144.8</td>
</tr>
<tr>
<td>G.729</td>
<td>128</td>
<td>20</td>
<td>34.4</td>
<td>39.2</td>
<td>72.4</td>
</tr>
<tr>
<td>G.729</td>
<td>128</td>
<td>30</td>
<td>25.6</td>
<td>30.4</td>
<td>52.53</td>
</tr>
<tr>
<td>G.729</td>
<td>128</td>
<td>40</td>
<td>21.2</td>
<td>26</td>
<td>42.6</td>
</tr>
<tr>
<td>G.729</td>
<td>128</td>
<td>50</td>
<td>18.56</td>
<td>20.8</td>
<td>34.08</td>
</tr>
<tr>
<td>G.729</td>
<td>128</td>
<td>60</td>
<td>16.8</td>
<td>19.47</td>
<td>30.53</td>
</tr>
<tr>
<td>G.729</td>
<td>64</td>
<td>10</td>
<td>60.8</td>
<td>72</td>
<td>138.8</td>
</tr>
<tr>
<td>G.729</td>
<td>64</td>
<td>20</td>
<td>34.4</td>
<td>39.2</td>
<td>72.4</td>
</tr>
<tr>
<td>G.729</td>
<td>64</td>
<td>30</td>
<td>25.6</td>
<td>30.4</td>
<td>52.53</td>
</tr>
<tr>
<td>G.729</td>
<td>64</td>
<td>40</td>
<td>21.2</td>
<td>24.4</td>
<td>41</td>
</tr>
<tr>
<td>G.729</td>
<td>64</td>
<td>50</td>
<td>18.56</td>
<td>20.8</td>
<td>34.08</td>
</tr>
<tr>
<td>G.729</td>
<td>64</td>
<td>60</td>
<td>16.8</td>
<td>18.4</td>
<td>29.47</td>
</tr>
</tbody>
</table>

\[ I_{E-\text{EF}} = I_E + (95 - I_E) \cdot \frac{P_{pl}}{P_{pl} + B_{pl}} \]  

\( I_{E-\text{EF}} \) is the so-called Burst Ratio, when packet loss is random \( \text{BurstR} = 1 \) and when packet loss is bursty \( \text{BurstR} > 1 \). For packet loss distributions corresponding to a 2-state Markov model with transition probabilities \( p \) between a “found” and a “loss” state, and \( q \) between the “loss” and the “found” state, the Burst Ratio can be calculated as:

\[ \text{BurstR} = \frac{1}{p + q} \]  

Once the \( I_{E-\text{EF}} \) factor is calculated, it is not difficult to determine R-factor as an output of E-Model using implicit values recommended in ITU-T G.107, which are \( R_0 = 94.7688 \), \( I_S = 1.4136 \) and \( A = 0 \). Hence Equation 9 can be modified to:

\[ R = 93.3553 - I_D - I_{E-\text{EF}} \]
The model used to estimate $I_D$ is described in [12], where it is explained that the effects of delay are well-known and easily modelled. Delays of less than 175 ms have a small effect on conversational difficulty, then $I_D = 4T$ where $T$ is the delay in ms.

7 Conclusion

Real-time applications are very sensitive to packet loss, and each variation occurring on the network can modify and influence the final result of a real-time data transmission, such as a VoIP call. On the one hand, the defence techniques using cryptography such as OpenVPN reduce the danger of security threats but on the other hand, they affect the required bandwidth of IP telephony which is significantly increased in the case of OpenVPN. The relations presented in this paper help us to understand how OpenVPN and TLS can affect the bandwidth of calls and how we can optimize timing.

For example, we can show an optimization at G.723.1 with 6.3 kbps, see Table 1. If we used a timing of 30 ms during packetization, we would require 30.4 kbps in the case of TLS, against 24 kbps in an environment without TLS, but we could achieve a better result with a timing of 60 ms, because we would require only 17.33 kbps for TLS against 15.2 kbps without TLS and it is really a much better ratio. This presented example is valid for AES due to the size of the blocks but the newly created relations help to optimize any CBC encryption.

The new contribution of this paper is the presented method of bandwidth calculation in a network using TLS. The achieved results were confirmed on a testbed, the bandwidth of any particular call was affected by the length of cipher blocks and didn’t depend on key size. The results corresponded with relations stated in chapter 4. This paper is an extension of a previous work on the impact of security on the quality of VoIP calls [2] and [9].

I would like to thank my colleagues Antonio Nappa and Alessandro Rozza from the Milan University for collaboration and especially Filip Řezáč, a student of the VŠB – Technical University of Ostrava, who helped me with configuration of TLS.

References


Part III

Middleware and Applications
Time-Stamp Authority

VLADIMÍR SMOTLACHA, MILAN SOVA

Abstract
This document describes the time-stamp authority (TSA) that we have designed and built with a focus on providing accurate and trustworthy time information and on immunity from service compromising. Our system is based on open software and utilizes a hardware security module (HSM) which keeps the private key and speeds up cryptographic operations. We also used special hardware supporting clock synchronization and calibration.

Keywords: time-stamp authority, PKI, time synchronization

1 Time-Stamp Authority Principles

Time-stamp authority (TSA) is a service providing an electronic message containing a trustworthy time of issuance. The time-stamp protocol (TSP) is specified in RFC 3161 [1] defines both the communication protocol and the time-stamp format. The time information must be provided with a resolution of 1 second or finer. The TSP is based on a request message sent by a client (Time Stamp Query – TSQ) and a response message (Time Stamp Reply – TSR) sent by the server. The TSQ can contain arbitrary information (e.g. one-way hash of a file), which is returned back after being signed in the TSR. This way, the information is bound with the time stamp. The TSA clock must be synchronized by a trustworthy method to any time system that ensures traceability to UTC. According to the RFC3161, the TSP can be implemented either on the 3rd network level (TCP) or on the 4th network level (e.g. HTTP or SMTP).

Although there exists an open source implementation OpenTSA [2], the majority of operating TSAs are commercial proprietary solutions and therefore it is very difficult even for the operator to provide evidence of TSA time service accuracy. In some countries, laws force every licensed TSA operator to calibrate its TSA system. We will further discuss the issue in Section 5.

2 TSA Hardware

The TSA runs on a standard server with a 64-bit Intel processor. The hardware includes specialized cards that support data encryption and system clock synchronization. CPU performance is not critical, however the data encryption card requires a 64-bit operating system.
2.1 Hardware Security Module

In our application, the hardware security module serves mainly for safe keeping of the private key. We decided to use an SCA6000 card\(^1\) from Sun Microsystems – we already successfully utilized it in the CESNET Certification Authority. This card also accelerates cryptographic operations (up to 13000 RSA operations per second) therefore time stamp signing is not a bottleneck of our TSA. However, more important is the system immunity from service compromising as the private key is neither stored in the system memory nor used by the CPU. If an intruder hacks the operating system or even steals the computer, he has no way to read the private key.

2.2 Clock

The system is also equipped with the PCT-7424 card\(^2\) from Tedia s.r.o., which processes the incoming PPS (pulse-per-second) signal without interrupt latency. The PPS signal represents an external time source which is used by the NTP daemon for system clock synchronization. The clock stability is further improved by the ovenized oscillator which replaces the motherboard crystal 14.318 MHz.

3 TSA Software

The SCA6000 card is Sun Microsystems proprietary hardware and the manufacturer provides a driver only for Solaris and Linux. We decided to use Linux but only a limited number of distributions and kernels is supported – the newest one is RedHat 5.0 with x86_64 kernel version 2.6.9-22.

The whole TSA functionality is implemented in the OpenTSA package\(^2\) which includes:

- a patch enhancing the OpenSSL by the time-stamp protocol according to RFC-3161
- the Apache2 module that implements the service on the HTTP protocol

The OpenTSA package unfortunately lacks the SCA6000 card support and we were forced to program this interface ourselves.

Another important system program is the NTP daemon (we have installed recent version 4.2.4) which is responsible for TSA clock synchronization.

---


4 Clock Synchronization

The basic assumption of every time stamp is that the issuing TSA guarantees a declared level of internal clock accuracy. The TSA operator must arrange the clock synchronization by a trustworthy method to any time system that ensures traceability to UTC. The most common method is to utilize a GPS receiver generating the PPS signal. Alternatively, another PPS source can be used as well, e.g. a Caesium clock. The PPS signal from our GPS Trimble Acutime 2000 has a maximal time error of 50 nanoseconds and the PPS capture card PCT-7424 adds an uncertainty of 50 nanoseconds, too. Passing the PCI bus and further software processing increases the total uncertainty to about 300 nanoseconds. The running NTP daemon disciplines the system clock which represents the time scale T(TSA).

TSA clock stability is about $10^{-9}$ – it depends on the ovenized oscillator (OCXO). Even when no PPS signal is available for a day, the TSA time uncertainty is still better than 100 microseconds.

We developed a simple method that allows one to calibrate the system clock: the utility gen_pps transmits $T(TSA)$ to a serial port in the form of a generated PPS signal and this signal is compared with the reference clock. Long term measurement proved that we synchronized the time scale of our TSA prototype with an uncertainty of less than 2 microseconds. Figure 1 shows an example of such a one-day measurement.

The described method of system clock synchronization was adopted from our design of CESNET primary NTP servers [5].

5 TSA Calibration

TSA provides time information with a specified accuracy. As any other time service, it should be subjected to calibration that allows tracing of the local time scale to UTC as requested in time metrology. However, TSA calibration is a new topic – we designed one of such methods in [3] and further elaborated it in [4].

TSA issues a time stamp that binds an event (the client’s request TSQ) with an epoch with some uncertainty which includes:

- Time offset of the TSA clock: It is the primary source of inaccuracy and it has to be considered in the TSA service specification. Accuracy of the TSA clock has been discussed in Section 4.
- Delay in TSQ and TSR processing: The delay summarizes particular delays inside the TSA system, including client authorization and authentication, response generation and queueing of network interfaces. This delay is mainly done by the TSA software design and also may depend on the instantaneous TSA load.
Network transport delay of the TSQ from client to service provider: The network delay (and the delay variation) significantly influences the overall TSA accuracy. While the mean network delay can be easily measured, it is difficult to estimate the maximum network delay.

It is evident that the service provider cannot guarantee any parameters that depend on the network between service provider and service client. While it is important and meaningful to evaluate the service accuracy at the particular client site, real calibration is possible only at the provider site.

TSA calibration is based on evaluation of the difference between issued time stamp and the epoch at which the TSQ occurred at the reference point (e.g. TSA network interface, network border router) specified by the service provider.

Time stamps are generated from the server time scale $T(S)$ synchronized to UTC with an uncertainty $\pm u(S)$. The uncertainty $u(TS)$ of the time stamps is in principle larger and practically much larger than $u(S)$ due to limited resolution of the time stamps (RFC 3161 [1] states that $u(TS)$ can be up to 1 s). Uncertainties
u(S) and u(TS) are sometimes confused with the uncertainty u(UTC) of the local source of UTC – typically

\[ u(TS) \gg u(S) \gg u(UTC) \].

The basic calibration objective is to verify the uncertainty u(TS) which is associated with the access point defined by the TSA provider (typically at the server interface).

The TSA service calibration is carried out by the Calibration Computer (CC) which is described in [3]. We connected the CC to the same segment of LAN as the TSA and made the active calibration which employs the CC as a substitute for the TSA client as shown in Figure 2.

**Figure 2.** Method of TSA calibration

Let the CC clock represents the time scale \( T(C) \) and TSA clock represents the time scale \( T(S) \). The TSQ is sent by the CC at time \( T_q(C) \) (with uncertainty \( u(Q) \)) and the TSR that contains the time stamp \( T_s(S) \) is received by the CC at time \( T_r(C) \). The uncertainty of the time stamp is \( u(TS) \). The time scale difference is

\[ x_0 = T_s(S) - T_q(C) \]

where the uncertainty of \( x_0 \) is given by \( u(TS) \) since \( u(Q) \ll u(TS) \).

The `tsa_mon` utility evaluates times \( T_q(C) \) and \( T_r(C) \), matches corresponding TSQ and TSR, and decodes the TS(S).

Values \( T_s - T_q \) in Figure 3 show the calibration result of our TSA. For comparison, we also evaluated the total response time \( T_r - T_q \).

We see that the mean value of \( T_s - T_q \) is 0.8 millisecond. Standard deviation of shown data is 0.16 milliseconds.
6 System Utilization

From the user viewpoint, obtaining a time stamp consists of several steps:
- TSQ generation
- time stamp request, i.e. sending a TSQ to the TSA
- TSR receiving and optional parsing

6.1 TS Command

The OpenTSA package contains a tool ts, which is accessed as a new openssl command. The ts has three subcommands:

`query`

The query subcommand generates a TSQ. It is used by the TSA client.

`reply`

The reply subcommand generates the TSR as a response to the TSQ. It is
used by the TSA server (in standard implementation with HTTP protocol, it is executed by the Apache module).

**verify**

The `verify` subcommand checks the TSR integrity. It is used by the client or any other third party to check if the received TSR matches the TSQ.

Example:

```bash
openssl ts -query -data design1.txt -no_nonce -out design1.tsq
```

OpenTSA web pages contain a detailed description of ts3.

### 6.2 Time Stamp Requesting

The URL of our TSA is `http://tsa.cesnet.cz:3161/tsa`, which means that it is accessible by the HTTP protocol. The TSA request can be made by the following command:

```bash
```

assuming that `design1.tsq` is the TSQ file generated in the previous example and TSR is stored as `dat.tsr`.

We also compiled a static executable `ts` binary (32-bit Linux), that can be simply installed without the need of patched OpenSSL build. The package contains scripts for arbitrary file time-stamping and time stamp verification. Until we finish the official CESNET TSA web pages, the package can be downloaded from a temporary repository4.

### 7 System Performance and Security Considerations

The system is supposed to provide accurate, reliable, and trustworthy signatures. To achieve this goal, the security of the signing key is essential. The signing key is generated within an HSM as an unexportable object, i.e. it can never leave the HSM. The key is activated by an operator by entering an activation pass phrase on every start of the system. Thus the only process able to use the key is the TSA server, even the root account cannot use the key without the pass phrase.

To mitigate a possible attack involving modification of the system time, the time is being constantly monitored by the NTPMON system [6]. Any significant

---


time discrepancy would invoke an inquiry into the system with the possibility of revoking the TSA certificate and effectively revoking the time-stamps issued within the respective time interval.

The system is naturally vulnerable to denial-of-service attack. However, our stress tests proved the throughput of almost 3000 time stamps per second. In case of need, the system and the firewall in front of it may be configured to limit the service to authenticated clients from an authorized network only.

References


---

5 http://www.ietf.org/rfc/rfc3161.txt
6 http://www.opentsa.org/site/
eduroam Authentication over Jammed Network

Jan Tomášek, Milan Sova

Abstract

Current eduroam can use several different transport mechanisms to carry messages of different EAP authentication mechanisms. Any particular combination is susceptible to disruption of the transport network at some level. This paper compares the resistance of some of the commonly used configurations to network disruptions.

Keywords: RADIUS, EAP, PEAP, eduroam

1 eduroam Authentication Methods

eduroam\(^1\) is a wireless roaming network where the user's authentication is provided by his home institution. In the current implementation, this is achieved by proxying RADIUS packets from the access point in the visited network to the RADIUS server at the home institution. The "classic" UDP-based RADIUS transport is susceptible to network disruptions between the visited and home networks.

RadSec ("RADIUS over TLS", see [1]) has been recently tested. Using TCP for transport, RadSec's response to packet loss is different from the plain RADIUS.

The most common authentication mechanisms used in eduroam are PEAP-MSCHAPv2, EAP-TTLS and EAP-TLS. The first two are password-based, the last one uses X.509 certificates to authenticate clients (servers are authenticated by X.509 certificates in all three cases). The protocol exchange in PEAP-MSCHAPv2 and EAP-TTLS is almost identical, we have therefore chosen to consider only the former for the sake of simplicity. PEAP-MSCHAPv2 requires the exchange of 11 RADIUS Request-Response packets.

EAP-TLS requires 6 Request-Response exchanges. In addition to that, in contrast to the password-based authentication methods, EAP-TLS does not in principle require contacting the authentication server within the user’s home institution - a client certificate can be just as easily verified by a service in the visited institution. This configuration obviously eliminates any effect on the quality of the link between the visited and home institution and can be considered as the reference for the other methods.

\(^1\) http://www.eduroam.org/

L. Lhostka, P. Satrapi (Editors) – Networking Studies III: Selected Technical Reports, p. 135–148
2 Measuring Setup

We used the following infrastructure to measure the influence of network quality on the user experience during authentication:

**server 1 - RADIUS server**
- Fujitsu-Siemens RX100, 1x Intel Pentium D @ 2.80GHz, 3GB RAM, 1Gbps Ethernet; Debian GNU/Linux Etch, Radiator 4.2 + patch 1.904

**server 2 - RADIUS server**
- Supermicro, 2x Intel Pentium 4 @ 3.20GHz, 2GB RAM, 1Gbps Ethernet; Debian GNU/Linux Etch, Radiator 4.2 + patch 1.904

**server 3 - RADIUS server**
- DELL PE1750, 2x Intel Xeon @ 3.06GHz, 2GB RAM, 1Gbps Ethernet; Debian GNU/Linux Etch, Radiator 4.2 + patch 1.904

**server 4 - RADIUS server**
- DELL PE2950, 2x Intel Xeon X5355 @ 2.66GHz, 16GB RAM, 1Gbps Ethernet; Debian GNU/Linux Etch, Radiator 4.2 + patch 1.904

**jammer - a router with defined packet loss**
- VIA Nehemiah @ 1GHz, 512 MB RAM, 100Mbps Ethernet; Slackware Linux 10.0.0, NIST Net 2.0.12b²

We connected the servers to a private network (see Figure 1 and Figure 2). The client (eapol_test from the wpa_supplicant package) was connected directly to the “visited” RADIUS server via ethernet (no access point involved). The tests over WiFi provided a dispersion too high to be useful (some values are provided in Section 4).

3 The Effect of Multiple RADIUS Proxies

The first measurement served for calibrating the whole infrastructure. The systems were inter-connected as in Figure 1. We tested both authentication methods (EAP-TLS, PEAP-MSCHAPv2) over both transports (RADIUS, RadSec).

The delay caused by connecting individual RADIUS servers is recorded in Table 1 for EAP-TLS and in Table 2 for PEAP-MSCHAPv2. For this measurement, the NIST Net router was configured to induce no packet loss.

The results show that one RADIUS server adds 0.05 seconds to the overall delay for EAP-TLS and 0.08 seconds for PEAP-MSCHAPv2. The difference is

eduroam Authentication over Jammed Network

Figure 1. Multiple RADIUS proxies

Table 1. The delay caused by RADIUS servers using EAP-TLS

<table>
<thead>
<tr>
<th>Transport</th>
<th>Servers</th>
<th>Success</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>1</td>
<td>100</td>
<td>0.06</td>
<td>0.06</td>
<td>0.01</td>
<td>0.05</td>
<td>0.12</td>
</tr>
<tr>
<td>UDP</td>
<td>1,2</td>
<td>100</td>
<td>0.11</td>
<td>0.11</td>
<td>0.01</td>
<td>0.11</td>
<td>0.18</td>
</tr>
<tr>
<td>UDP</td>
<td>1,2,3</td>
<td>100</td>
<td>0.16</td>
<td>0.16</td>
<td>0.01</td>
<td>0.15</td>
<td>0.23</td>
</tr>
<tr>
<td>RadSec</td>
<td>1,2</td>
<td>100</td>
<td>0.11</td>
<td>0.11</td>
<td>0.01</td>
<td>0.11</td>
<td>0.19</td>
</tr>
<tr>
<td>RadSec</td>
<td>1,2,3</td>
<td>100</td>
<td>0.16</td>
<td>0.16</td>
<td>0.00</td>
<td>0.15</td>
<td>0.22</td>
</tr>
</tbody>
</table>

Table 2. The delay caused by RADIUS servers using PEAP-MSCHAPv2

<table>
<thead>
<tr>
<th>Transport</th>
<th>Servers</th>
<th>Success</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>1</td>
<td>100</td>
<td>0.09</td>
<td>0.09</td>
<td>0.01</td>
<td>0.09</td>
<td>0.25</td>
</tr>
<tr>
<td>UDP</td>
<td>1,2</td>
<td>100</td>
<td>0.18</td>
<td>0.18</td>
<td>0.02</td>
<td>0.17</td>
<td>0.42</td>
</tr>
<tr>
<td>UDP</td>
<td>1,2,3</td>
<td>100</td>
<td>0.26</td>
<td>0.26</td>
<td>0.01</td>
<td>0.25</td>
<td>0.44</td>
</tr>
<tr>
<td>RadSec</td>
<td>1,2</td>
<td>100</td>
<td>0.18</td>
<td>0.18</td>
<td>0.01</td>
<td>0.17</td>
<td>0.28</td>
</tr>
<tr>
<td>RadSec</td>
<td>1,2,3</td>
<td>100</td>
<td>0.26</td>
<td>0.26</td>
<td>0.01</td>
<td>0.25</td>
<td>0.34</td>
</tr>
</tbody>
</table>

cauced by the fact that EAP-TLS requires 6 Request-Response exchanges while PEAP-MSCHAPv2 needs 11 of them.

Note that NIST Net router does not add to the delay.
4 The Effect of Wireless Connection

The tables in this section describe mainly the influence of a wireless channel to the precision of the measurement.

Table 3. The delay caused by RADIUS servers using WiFi 802.11b (2.4GHz) and EAP-TLS

<table>
<thead>
<tr>
<th>Transport</th>
<th>Servers</th>
<th>Success</th>
<th>Mean</th>
<th>Median</th>
<th>Std. dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>1</td>
<td>100</td>
<td>3.42</td>
<td>3.14</td>
<td>0.85</td>
<td>0.30</td>
<td>6.17</td>
</tr>
<tr>
<td>UDP</td>
<td>1,2</td>
<td>100</td>
<td>4.22</td>
<td>3.25</td>
<td>5.13</td>
<td>0.30</td>
<td>75.42</td>
</tr>
<tr>
<td>UDP</td>
<td>1,2,3</td>
<td>98</td>
<td>4.62</td>
<td>3.26</td>
<td>6.19</td>
<td>0.41</td>
<td>70.43</td>
</tr>
</tbody>
</table>

Table 4. The delay caused by RADIUS servers using WiFi 802.11a (5GHz) and EAP-TLS

<table>
<thead>
<tr>
<th>Transport</th>
<th>Servers</th>
<th>Success</th>
<th>Mean</th>
<th>Median</th>
<th>Std. dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>1</td>
<td>100</td>
<td>4.81</td>
<td>4.61</td>
<td>2.16</td>
<td>4.50</td>
<td>41.87</td>
</tr>
<tr>
<td>UDP</td>
<td>1,2</td>
<td>99</td>
<td>4.98</td>
<td>4.62</td>
<td>3.04</td>
<td>4.51</td>
<td>41.84</td>
</tr>
<tr>
<td>UDP</td>
<td>1,2,3</td>
<td>100</td>
<td>4.92</td>
<td>4.71</td>
<td>2.16</td>
<td>4.61</td>
<td>41.85</td>
</tr>
</tbody>
</table>

Note the high variance of the time needed to authenticate. This precludes using the wireless channel for measuring the robustness of different authentication methods with respect to the transport channel quality.

5 The Effect of Packet Loss

To find out what influence packet loss has on the eduroam authentication process, we set up the systems according to Figure 2. A usual wpa_supplicant’s strategy was utilized by the client based on re-sending requests until either the connection is established or the authentication timer (120 seconds in our case) expires.

We tested the responsiveness of individual authentication mechanisms over different transports in three scenarios:

1. the NIST Net router dropped packets going from the client to the RADIUS server
2. the NIST Net router dropped packets going from the RADIUS server to the client
3. the NIST Net router dropped packets going in both directions

The results are provided in the following sections where the following terms are defined:

- **Loss**
  - packet loss

- **Success**
  - rate of successful authentications

- **Mean**
  - mean time before successful authentication

- **Median**
  - median time before successful authentication

- **Std. dev.**
  - standard deviation of the time before successful authentication

- **Min**
  - minimal value of the time before successful authentication

- **Max**
  - maximal value of the time before successful authentication
5.1 Dropping Packets from client to server1

For the first set of measurements, the NIST Net router discarded packets originating at server2 designated for server1.

Table 5. EAP-TLS over UDP, packet loss from client to server

<table>
<thead>
<tr>
<th>Loss [%]</th>
<th>Success [%]</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>100</td>
<td>0.28</td>
<td>0.11</td>
<td>0.69</td>
<td>0.11</td>
<td>3.12</td>
</tr>
<tr>
<td>5.0</td>
<td>100</td>
<td>1.31</td>
<td>0.11</td>
<td>2.96</td>
<td>0.11</td>
<td>24.12</td>
</tr>
<tr>
<td>10.0</td>
<td>100</td>
<td>3.27</td>
<td>0.11</td>
<td>6.14</td>
<td>0.11</td>
<td>42.13</td>
</tr>
<tr>
<td>20.0</td>
<td>99</td>
<td>6.71</td>
<td>3.11</td>
<td>9.33</td>
<td>0.11</td>
<td>48.15</td>
</tr>
<tr>
<td>30.0</td>
<td>96</td>
<td>15.59</td>
<td>9.11</td>
<td>15.18</td>
<td>0.11</td>
<td>90.17</td>
</tr>
<tr>
<td>40.0</td>
<td>90</td>
<td>24.34</td>
<td>24.12</td>
<td>19.70</td>
<td>0.11</td>
<td>87.14</td>
</tr>
<tr>
<td>50.0</td>
<td>68</td>
<td>35.22</td>
<td>30.13</td>
<td>22.34</td>
<td>0.11</td>
<td>96.14</td>
</tr>
<tr>
<td>60.0</td>
<td>49</td>
<td>42.13</td>
<td>45.13</td>
<td>27.08</td>
<td>0.11</td>
<td>114.16</td>
</tr>
<tr>
<td>70.0</td>
<td>19</td>
<td>48.39</td>
<td>48.13</td>
<td>24.43</td>
<td>0.11</td>
<td>117.17</td>
</tr>
<tr>
<td>80.0</td>
<td>3</td>
<td>75.15</td>
<td>72.15</td>
<td>20.62</td>
<td>0.11</td>
<td>111.16</td>
</tr>
</tbody>
</table>

Table 6. EAP-TLS over RadSec, packet loss from client to server

<table>
<thead>
<tr>
<th>Loss [%]</th>
<th>Success [%]</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>100</td>
<td>0.13</td>
<td>0.11</td>
<td>0.06</td>
<td>0.11</td>
<td>0.32</td>
</tr>
<tr>
<td>5.0</td>
<td>100</td>
<td>0.19</td>
<td>0.11</td>
<td>0.16</td>
<td>0.11</td>
<td>1.56</td>
</tr>
<tr>
<td>10.0</td>
<td>100</td>
<td>0.25</td>
<td>0.11</td>
<td>0.21</td>
<td>0.11</td>
<td>1.57</td>
</tr>
<tr>
<td>20.0</td>
<td>100</td>
<td>0.54</td>
<td>0.32</td>
<td>0.58</td>
<td>0.11</td>
<td>3.64</td>
</tr>
<tr>
<td>30.0</td>
<td>98</td>
<td>0.86</td>
<td>0.53</td>
<td>0.80</td>
<td>0.11</td>
<td>4.48</td>
</tr>
<tr>
<td>40.0</td>
<td>97</td>
<td>1.53</td>
<td>0.94</td>
<td>2.66</td>
<td>0.11</td>
<td>22.95</td>
</tr>
<tr>
<td>50.0</td>
<td>80</td>
<td>3.91</td>
<td>1.98</td>
<td>7.77</td>
<td>0.11</td>
<td>50.53</td>
</tr>
<tr>
<td>60.0</td>
<td>57</td>
<td>5.12</td>
<td>2.51</td>
<td>8.01</td>
<td>0.32</td>
<td>50.13</td>
</tr>
<tr>
<td>70.0</td>
<td>20</td>
<td>16.51</td>
<td>7.18</td>
<td>17.24</td>
<td>0.32</td>
<td>51.98</td>
</tr>
<tr>
<td>80.0</td>
<td>1</td>
<td>33.01</td>
<td>25.70</td>
<td>nan</td>
<td>24.45</td>
<td>48.87</td>
</tr>
</tbody>
</table>
Table 7. PEAP-MSCHAPv2 over UDP, packet loss from client to server

<table>
<thead>
<tr>
<th>Loss [%]</th>
<th>Success</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>100</td>
<td>0.40</td>
<td>0.18</td>
<td>0.82</td>
<td>0.17</td>
<td>6.18</td>
</tr>
<tr>
<td>5.0</td>
<td>100</td>
<td>1.97</td>
<td>0.18</td>
<td>3.01</td>
<td>0.17</td>
<td>24.19</td>
</tr>
<tr>
<td>10.0</td>
<td>100</td>
<td>6.16</td>
<td>3.18</td>
<td>8.10</td>
<td>0.17</td>
<td>66.20</td>
</tr>
<tr>
<td>20.0</td>
<td>98</td>
<td>13.64</td>
<td>6.18</td>
<td>13.20</td>
<td>0.17</td>
<td>69.20</td>
</tr>
<tr>
<td>30.0</td>
<td>92</td>
<td>27.16</td>
<td>24.20</td>
<td>20.86</td>
<td>0.17</td>
<td>96.22</td>
</tr>
<tr>
<td>40.0</td>
<td>78</td>
<td>40.03</td>
<td>33.20</td>
<td>24.30</td>
<td>0.17</td>
<td>111.24</td>
</tr>
<tr>
<td>50.0</td>
<td>53</td>
<td>56.02</td>
<td>54.20</td>
<td>21.88</td>
<td>9.18</td>
<td>108.23</td>
</tr>
<tr>
<td>60.0</td>
<td>17</td>
<td>70.74</td>
<td>75.21</td>
<td>28.07</td>
<td>6.17</td>
<td>117.24</td>
</tr>
<tr>
<td>70.0</td>
<td>5</td>
<td>87.82</td>
<td>93.21</td>
<td>19.02</td>
<td>51.20</td>
<td>105.22</td>
</tr>
<tr>
<td>80.0</td>
<td>0</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
</tr>
</tbody>
</table>

Table 8. PEAP-MSCHAPv2 over RadSec, packet loss from client to server

<table>
<thead>
<tr>
<th>Loss [%]</th>
<th>Success</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>100</td>
<td>0.20</td>
<td>0.18</td>
<td>0.07</td>
<td>0.18</td>
<td>0.60</td>
</tr>
<tr>
<td>5.0</td>
<td>100</td>
<td>0.29</td>
<td>0.18</td>
<td>0.16</td>
<td>0.17</td>
<td>1.01</td>
</tr>
<tr>
<td>10.0</td>
<td>100</td>
<td>0.45</td>
<td>0.39</td>
<td>0.31</td>
<td>0.18</td>
<td>1.66</td>
</tr>
<tr>
<td>20.0</td>
<td>99</td>
<td>0.91</td>
<td>0.80</td>
<td>0.69</td>
<td>0.18</td>
<td>4.75</td>
</tr>
<tr>
<td>30.0</td>
<td>97</td>
<td>1.71</td>
<td>1.43</td>
<td>1.67</td>
<td>0.18</td>
<td>22.84</td>
</tr>
<tr>
<td>40.0</td>
<td>91</td>
<td>3.08</td>
<td>2.26</td>
<td>4.02</td>
<td>0.18</td>
<td>50.82</td>
</tr>
<tr>
<td>50.0</td>
<td>74</td>
<td>5.76</td>
<td>3.74</td>
<td>8.42</td>
<td>0.38</td>
<td>51.55</td>
</tr>
<tr>
<td>60.0</td>
<td>34</td>
<td>10.71</td>
<td>5.99</td>
<td>12.14</td>
<td>0.80</td>
<td>54.27</td>
</tr>
<tr>
<td>70.0</td>
<td>8</td>
<td>18.51</td>
<td>9.01</td>
<td>18.54</td>
<td>2.06</td>
<td>72.20</td>
</tr>
<tr>
<td>80.0</td>
<td>0</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
</tr>
</tbody>
</table>

5.2 Dropping Packets from server1 to client

In the second set of measurement, the NIST Net router discarded packets originating at server1 and designated for server2.

5.3 Dropping Packets in Both Directions

The packet loss values in the tables and graphs in this section were configured for both directions. I. e. the overall packet loss on the link was twice as high as the value listed. For instance, a value of 20% means 20% of packets lost from client to server plus another 20% lost in the opposite direction.
6 Conclusions

TCP transport is significantly more reliable than UDP and provides an authentication success rate of more than 90% over links with a 40% packet loss. With worsening quality of the link the success rate decreases dramatically below 20% with 60% of packets lost.

The reliance of UDP transport on link quality is much more linear, going below the usability threshold of 50% at around 35% lost packets, depending on the length of authentication exchange. The more favourable results for client-to-server jamming are caused by the wpa_supplicant’s aggressive packet re-sending strategy compared to the behaviour of Radiator.

The measuring method used was quite simple. In addition to just dropping packets, real networks might misbehave with regards to the size of a packet (e.g. MTU problems) or the influence of other traffic. However, the responsiveness of TCP and UDP transport should be comparable to our results.
Figure 4. Dropping packets from server1 to client

Table 9. EAP-TLS over UDP, packet loss from server to client

<table>
<thead>
<tr>
<th>Loss [%]</th>
<th>Success [%]</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>100</td>
<td>0.23</td>
<td>0.11</td>
<td>0.68</td>
<td>0.11</td>
<td>6.10</td>
</tr>
<tr>
<td>5.0</td>
<td>98</td>
<td>1.02</td>
<td>0.11</td>
<td>1.58</td>
<td>0.11</td>
<td>9.12</td>
</tr>
<tr>
<td>10.0</td>
<td>94</td>
<td>1.72</td>
<td>0.11</td>
<td>2.78</td>
<td>0.11</td>
<td>27.12</td>
</tr>
<tr>
<td>20.0</td>
<td>80</td>
<td>4.44</td>
<td>3.11</td>
<td>6.00</td>
<td>0.11</td>
<td>33.11</td>
</tr>
<tr>
<td>30.0</td>
<td>61</td>
<td>6.89</td>
<td>3.11</td>
<td>9.17</td>
<td>0.11</td>
<td>51.12</td>
</tr>
<tr>
<td>40.0</td>
<td>46</td>
<td>8.77</td>
<td>6.10</td>
<td>9.44</td>
<td>0.11</td>
<td>57.12</td>
</tr>
<tr>
<td>50.0</td>
<td>25</td>
<td>12.77</td>
<td>6.11</td>
<td>12.41</td>
<td>0.11</td>
<td>54.11</td>
</tr>
<tr>
<td>60.0</td>
<td>7</td>
<td>13.71</td>
<td>6.11</td>
<td>14.35</td>
<td>0.11</td>
<td>54.11</td>
</tr>
<tr>
<td>70.0</td>
<td>3</td>
<td>24.77</td>
<td>27.11</td>
<td>13.26</td>
<td>6.10</td>
<td>51.12</td>
</tr>
<tr>
<td>80.0</td>
<td>0</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
</tr>
</tbody>
</table>
Table 10. EAP-TLS over RadSec, packet loss from server to client

<table>
<thead>
<tr>
<th>Loss [%]</th>
<th>Success</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>100</td>
<td>0.13</td>
<td>0.11</td>
<td>0.06</td>
<td>0.11</td>
<td>0.33</td>
</tr>
<tr>
<td>5.0</td>
<td>100</td>
<td>0.17</td>
<td>0.11</td>
<td>0.13</td>
<td>0.11</td>
<td>0.99</td>
</tr>
<tr>
<td>10.0</td>
<td>100</td>
<td>0.27</td>
<td>0.11</td>
<td>0.21</td>
<td>0.11</td>
<td>1.41</td>
</tr>
<tr>
<td>20.0</td>
<td>99</td>
<td>0.51</td>
<td>0.33</td>
<td>0.48</td>
<td>0.11</td>
<td>3.78</td>
</tr>
<tr>
<td>30.0</td>
<td>99</td>
<td>0.94</td>
<td>0.54</td>
<td>1.00</td>
<td>0.11</td>
<td>7.90</td>
</tr>
<tr>
<td>40.0</td>
<td>95</td>
<td>1.65</td>
<td>0.98</td>
<td>3.16</td>
<td>0.11</td>
<td>46.40</td>
</tr>
<tr>
<td>50.0</td>
<td>83</td>
<td>3.35</td>
<td>1.83</td>
<td>6.73</td>
<td>0.11</td>
<td>52.59</td>
</tr>
<tr>
<td>60.0</td>
<td>61</td>
<td>6.69</td>
<td>3.33</td>
<td>11.00</td>
<td>0.33</td>
<td>50.60</td>
</tr>
<tr>
<td>70.0</td>
<td>7</td>
<td>9.52</td>
<td>3.75</td>
<td>15.95</td>
<td>0.32</td>
<td>48.30</td>
</tr>
<tr>
<td>80.0</td>
<td>0</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
</tr>
</tbody>
</table>

Table 11. PEAP-MSCHAPv2 over UDP, packet loss from server to client

<table>
<thead>
<tr>
<th>Loss [%]</th>
<th>Success</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>100</td>
<td>0.60</td>
<td>0.18</td>
<td>1.12</td>
<td>0.17</td>
<td>6.17</td>
</tr>
<tr>
<td>5.0</td>
<td>98</td>
<td>1.34</td>
<td>0.17</td>
<td>1.90</td>
<td>0.17</td>
<td>6.18</td>
</tr>
<tr>
<td>10.0</td>
<td>88</td>
<td>4.67</td>
<td>3.17</td>
<td>4.93</td>
<td>0.17</td>
<td>24.21</td>
</tr>
<tr>
<td>20.0</td>
<td>65</td>
<td>9.98</td>
<td>6.17</td>
<td>9.86</td>
<td>0.17</td>
<td>54.18</td>
</tr>
<tr>
<td>30.0</td>
<td>37</td>
<td>13.79</td>
<td>9.17</td>
<td>10.41</td>
<td>0.17</td>
<td>36.17</td>
</tr>
<tr>
<td>40.0</td>
<td>20</td>
<td>21.77</td>
<td>13.66</td>
<td>15.44</td>
<td>6.16</td>
<td>57.19</td>
</tr>
<tr>
<td>50.0</td>
<td>10</td>
<td>28.07</td>
<td>28.68</td>
<td>12.52</td>
<td>12.15</td>
<td>54.18</td>
</tr>
<tr>
<td>60.0</td>
<td>5</td>
<td>18.82</td>
<td>12.16</td>
<td>12.09</td>
<td>6.16</td>
<td>36.19</td>
</tr>
<tr>
<td>70.0</td>
<td>1</td>
<td>27.20</td>
<td>27.20</td>
<td>nan</td>
<td>27.20</td>
<td>27.20</td>
</tr>
<tr>
<td>80.0</td>
<td>0</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
</tr>
</tbody>
</table>
Table 12. PEAP-MSCHAPv2 over RadSec, packet loss from server to client

<table>
<thead>
<tr>
<th>Loss [%]</th>
<th>Success</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>100</td>
<td>0.21</td>
<td>0.18</td>
<td>0.07</td>
<td>0.17</td>
<td>0.39</td>
</tr>
<tr>
<td>5.0</td>
<td>100</td>
<td>0.32</td>
<td>0.18</td>
<td>0.18</td>
<td>0.17</td>
<td>1.06</td>
</tr>
<tr>
<td>10.0</td>
<td>100</td>
<td>0.43</td>
<td>0.39</td>
<td>0.25</td>
<td>0.17</td>
<td>1.23</td>
</tr>
<tr>
<td>20.0</td>
<td>100</td>
<td>0.94</td>
<td>0.81</td>
<td>0.66</td>
<td>0.17</td>
<td>4.20</td>
</tr>
<tr>
<td>30.0</td>
<td>96</td>
<td>1.54</td>
<td>1.25</td>
<td>1.07</td>
<td>0.18</td>
<td>5.45</td>
</tr>
<tr>
<td>40.0</td>
<td>92</td>
<td>3.32</td>
<td>2.50</td>
<td>3.77</td>
<td>0.59</td>
<td>22.69</td>
</tr>
<tr>
<td>50.0</td>
<td>76</td>
<td>5.95</td>
<td>3.89</td>
<td>7.63</td>
<td>0.60</td>
<td>47.95</td>
</tr>
<tr>
<td>60.0</td>
<td>29</td>
<td>12.68</td>
<td>6.56</td>
<td>12.90</td>
<td>1.65</td>
<td>51.97</td>
</tr>
<tr>
<td>70.0</td>
<td>7</td>
<td>34.52</td>
<td>31.87</td>
<td>20.17</td>
<td>5.31</td>
<td>58.03</td>
</tr>
<tr>
<td>80.0</td>
<td>0</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
</tr>
</tbody>
</table>

Table 13. EAP-TLS over UDP, packet loss in both directions

<table>
<thead>
<tr>
<th>Loss [%]</th>
<th>Success</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>100</td>
<td>0.47</td>
<td>0.11</td>
<td>1.03</td>
<td>0.11</td>
<td>6.11</td>
</tr>
<tr>
<td>3.0</td>
<td>99</td>
<td>1.54</td>
<td>0.11</td>
<td>3.14</td>
<td>0.11</td>
<td>24.13</td>
</tr>
<tr>
<td>5.0</td>
<td>97</td>
<td>2.11</td>
<td>0.11</td>
<td>3.53</td>
<td>0.11</td>
<td>24.12</td>
</tr>
<tr>
<td>10.0</td>
<td>88</td>
<td>4.53</td>
<td>3.11</td>
<td>6.55</td>
<td>0.11</td>
<td>54.13</td>
</tr>
<tr>
<td>15.0</td>
<td>74</td>
<td>9.00</td>
<td>3.12</td>
<td>11.36</td>
<td>0.10</td>
<td>54.13</td>
</tr>
<tr>
<td>20.0</td>
<td>63</td>
<td>11.22</td>
<td>6.11</td>
<td>11.25</td>
<td>0.11</td>
<td>51.12</td>
</tr>
<tr>
<td>25.0</td>
<td>52</td>
<td>17.33</td>
<td>9.10</td>
<td>15.67</td>
<td>0.10</td>
<td>72.13</td>
</tr>
<tr>
<td>30.0</td>
<td>35</td>
<td>20.41</td>
<td>21.12</td>
<td>15.92</td>
<td>0.11</td>
<td>75.14</td>
</tr>
<tr>
<td>40.0</td>
<td>14</td>
<td>32.30</td>
<td>30.12</td>
<td>25.40</td>
<td>3.10</td>
<td>114.15</td>
</tr>
<tr>
<td>50.0</td>
<td>3</td>
<td>46.33</td>
<td>46.62</td>
<td>25.83</td>
<td>9.11</td>
<td>96.15</td>
</tr>
</tbody>
</table>
### Table 14. EAP-TLS over RadSec, packet loss in both directions

<table>
<thead>
<tr>
<th>Loss [%]</th>
<th>Success</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>100</td>
<td>0.14</td>
<td>0.11</td>
<td>0.09</td>
<td>0.11</td>
<td>0.60</td>
</tr>
<tr>
<td>3.0</td>
<td>100</td>
<td>0.18</td>
<td>0.11</td>
<td>0.13</td>
<td>0.11</td>
<td>0.79</td>
</tr>
<tr>
<td>5.0</td>
<td>100</td>
<td>0.25</td>
<td>0.11</td>
<td>0.19</td>
<td>0.11</td>
<td>1.24</td>
</tr>
<tr>
<td>10.0</td>
<td>100</td>
<td>0.47</td>
<td>0.34</td>
<td>0.41</td>
<td>0.11</td>
<td>2.99</td>
</tr>
<tr>
<td>15.0</td>
<td>100</td>
<td>0.85</td>
<td>0.64</td>
<td>0.74</td>
<td>0.10</td>
<td>4.66</td>
</tr>
<tr>
<td>20.0</td>
<td>98</td>
<td>1.58</td>
<td>1.18</td>
<td>1.41</td>
<td>0.11</td>
<td>8.17</td>
</tr>
<tr>
<td>25.0</td>
<td>87</td>
<td>3.15</td>
<td>1.86</td>
<td>6.19</td>
<td>0.10</td>
<td>53.03</td>
</tr>
<tr>
<td>30.0</td>
<td>0</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
</tr>
<tr>
<td>40.0</td>
<td>0</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
</tr>
<tr>
<td>50.0</td>
<td>0</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
</tr>
</tbody>
</table>

### Table 15. PEAP-MSCHAPv2 over UDP, packet loss in both directions

<table>
<thead>
<tr>
<th>Loss [%]</th>
<th>Success</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>99</td>
<td>1.04</td>
<td>0.18</td>
<td>1.94</td>
<td>0.17</td>
<td>21.20</td>
</tr>
<tr>
<td>3.0</td>
<td>98</td>
<td>2.30</td>
<td>0.18</td>
<td>3.83</td>
<td>0.17</td>
<td>24.19</td>
</tr>
<tr>
<td>5.0</td>
<td>93</td>
<td>4.27</td>
<td>3.17</td>
<td>5.44</td>
<td>0.17</td>
<td>30.19</td>
</tr>
<tr>
<td>10.0</td>
<td>80</td>
<td>7.32</td>
<td>6.17</td>
<td>7.64</td>
<td>0.17</td>
<td>42.19</td>
</tr>
<tr>
<td>15.0</td>
<td>61</td>
<td>14.83</td>
<td>9.17</td>
<td>12.81</td>
<td>0.17</td>
<td>51.18</td>
</tr>
<tr>
<td>20.0</td>
<td>37</td>
<td>21.75</td>
<td>15.19</td>
<td>17.62</td>
<td>0.17</td>
<td>78.21</td>
</tr>
<tr>
<td>25.0</td>
<td>26</td>
<td>27.53</td>
<td>27.18</td>
<td>21.19</td>
<td>0.17</td>
<td>78.20</td>
</tr>
<tr>
<td>30.0</td>
<td>12</td>
<td>40.29</td>
<td>39.17</td>
<td>25.20</td>
<td>6.16</td>
<td>96.22</td>
</tr>
<tr>
<td>40.0</td>
<td>1</td>
<td>64.39</td>
<td>63.18</td>
<td>nan</td>
<td>36.17</td>
<td>90.20</td>
</tr>
<tr>
<td>50.0</td>
<td>0</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
</tr>
</tbody>
</table>
Table 16. PEAP-MSCHAPv2 over RadSec, packet loss in both directions

<table>
<thead>
<tr>
<th>Loss [%]</th>
<th>Success</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>Std. dev. [s]</th>
<th>Min [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>100</td>
<td>0.22</td>
<td>0.18</td>
<td>0.09</td>
<td>0.18</td>
<td>0.80</td>
</tr>
<tr>
<td>3.0</td>
<td>100</td>
<td>0.34</td>
<td>0.38</td>
<td>0.19</td>
<td>0.18</td>
<td>1.11</td>
</tr>
<tr>
<td>5.0</td>
<td>100</td>
<td>0.49</td>
<td>0.40</td>
<td>0.32</td>
<td>0.18</td>
<td>2.38</td>
</tr>
<tr>
<td>10.0</td>
<td>100</td>
<td>0.82</td>
<td>0.63</td>
<td>0.60</td>
<td>0.18</td>
<td>3.94</td>
</tr>
<tr>
<td>15.0</td>
<td>98</td>
<td>1.61</td>
<td>1.08</td>
<td>3.04</td>
<td>0.18</td>
<td>30.16</td>
</tr>
<tr>
<td>20.0</td>
<td>94</td>
<td>3.28</td>
<td>2.55</td>
<td>3.08</td>
<td>0.18</td>
<td>25.37</td>
</tr>
<tr>
<td>25.0</td>
<td>56</td>
<td>5.90</td>
<td>3.50</td>
<td>7.55</td>
<td>0.19</td>
<td>47.44</td>
</tr>
<tr>
<td>30.0</td>
<td>13</td>
<td>14.60</td>
<td>6.31</td>
<td>23.33</td>
<td>0.95</td>
<td>100.74</td>
</tr>
<tr>
<td>40.0</td>
<td>0</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
</tr>
<tr>
<td>50.0</td>
<td>0</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
</tr>
</tbody>
</table>

Figure 5. Dropping packets in both directions
References


Implementation of DXT Compression for UltraGrid

IAN WESLEY-SMITH, MILOŠ LIŠKA, PETR HOLUB

Abstract
This report describes an implementation of DXT compression in UltraGrid, allowing for low latency, high definition multi-party videoconferencing requiring only 250 Mbps of bandwidth. This represents a substantial decrease compared to the uncompressed video stream, which requires 1.5 Gbps, while having a minimal impact on latency. The DXT implementation includes a new client-side display which, when using DXT compressed data, takes full advantage of the graphics card, resulting in minimal CPU utilization. This report includes experimental evaluation of end-to-end latencies, CPU load, and traffic profiles for both uncompressed and DXT-compressed video.

Keywords: high-definition video over IP, DXT compression, UltraGrid, latency, performance, network traffic profiles

1 Introduction

An uncompressed high-definition (HD) video transmission creates an unparalleled experience for participants of collaborative environments, providing both high image quality and low end-to-end latency of the transmission (considered in an end-to-end way, i.e. all the way from the camera to the display screen). Utilization of such an approach has been demonstrated by several teams around the world in recent years: The UltraGrid\(^1\) system provided by Pekins & Gharai [1], a modified version of UltraGrid by CESNET team [2], iHDTV systems\(^2\), or the HDTV over IP project by NTT Laboratories [3]. Although, as noted earlier, uncompressed HD video provides the best image quality and lowest latency, its high bandwidth utilization, about 1.5 Gbps, severely hinders deployment of systems utilizing it.

In order to facilitate adoption in non 10GbE environments, we have integrated low-latency DXT compression [4] into UltraGrid. There is a series of DXT compression schemes named DXT1 through DXT5 [5], of which DXT1 provides the best compression ratio and thus has been actually used in UltraGrid. It is based on colour indexing, where each 4×4 pixel block (so called texel) is converted

---

\(^{1}\) http://ultragrid.east.isi.edu/

\(^{2}\) http://ihdtv.sourceforge.net/
into two 16-bit colours\(^3\) \((c_1, c_2)\) and a lookup table comprising 16 colour indices with 2 b per index. The index points to one of the following four colours then: \((c_1, 2/3c_1+1/3c_2, 1/3c_1+2/3c_2, c_2)\). Choice of the colours \((c_1, c_2)\) is up to the implementation and is an important factor for quality of resulting images. Discussion of alpha channel handling has been omitted in the discussion of DXT as it is not used in UltraGrid. DXT compression was selected for the following reasons:

- 6:1 bit rate reduction (by converting 4×4 pixel block with 24 b/pixel into 4×16 b structure; coincidentally, the same 6:1 ratio is also achieved when comparing the original 10 b 4:2:2 HD-SDI stream at 1.5 Gbps to the resulting 8 b 4:4:4 stream at 250 Mbps).
- acceptable image quality on natural scenes (computer-generated images with long fine gradients may display more significant posterization).
- availability of an implementation in real-time [6] and for HD frame size and rate (FastDXT library\(^4\)).
- availability of DXT texture decompression and rendering directly in graphics cards, even in low-cost models.

While the DXT compression has significant computational requirements, and thus requires a powerful enough sender computer, the receiver can be a fairly low cost computer equipped with a graphics card supporting DXT textures. The resulting data streams can be run over Gigabit Ethernet, which is now often available from LANs.

Implementation of JPEG2000 was also attempted in order to achieve a higher-quality compressed video, ideally with GPU support for both encoding and decoding as JPEG2000 compression and decompression are fairly exhaustive processes when implemented on a CPU only. This, however, proved to be infeasible due to the high read-back latency from the GPU using Cg\(^5\) (and CUDA\(^6\) only became available at the time of development). An overview of the measured latency is shown in Figures 1 and 2, including both component-wise latency and the overall latency of the whole process. The measurements were performed using a NVidia 7600GS (with both 9000-series driver and the new 100.14.11 driver) and a NVidia 8800GTX (with the 100.14.11 driver only). The results suggest that NVidia 8800-series cards are much better suited for general purpose computing compared to the older series, but still not usable for real-time compression using the Cg approach (this may significantly improve with the CUDA approach).

---

\(^3\) Each colour uses 5 b for red, 6 b for green, and 5 b for blue, i.e. 16-bit RGB 5:6:5.

\(^4\) [http://www.evl.uic.edu/cavern/fastdxt/](http://www.evl.uic.edu/cavern/fastdxt/)


Figure 1. Latency measurements of discrete wavelet transform performed on a GPU.
2 Implementation

The implementation requires two parts: the sender, where compression of the incoming data is done, and the receiver, where decompression happens as a part of the rendering process. The basic assumption is that the data will be sent to more than one receiver (be it for 1 : N distribution as in a virtual class, or full N : N collaboration where there are N – 1 receivers for each sender) and thus the playback part has to be as affordable as possible (both in terms of required CPU capacity and availability of required hardware). The sending part should also be
Implementation of DXT Compression for UltraGrid

optimized, but it is less critical compared to the receiving part. Another critical requirement is low end-to-end latency of the system, and as such the compression should introduce as little additional latency as possible. With the advent of general purpose GPU\textsuperscript{7} computing, we wanted to utilize the GPU’s computational power to the maximum extent available at the time of development.

2.1 Sender

We have opted to use the FastDXT library\textsuperscript{8} [7], which performs parallel compression of the data using CPUs. This library has been developed for compressing 4K video and can handle HD video in real time. It uses highly optimized code, based on intrinsics, resulting in efficient code generation across various platforms.

The UltraGrid sender acquires the video through one of its capture interfaces (DVS SDK for HDstation, Centaurus, Centaurus II and Centaurus II LT cards on Linux, or very flexible Quickɓme interface that allows for using various underlying capture cards). The acquired video data is extrapolated from 4:2:2 with 10 b per component to 4:4:4 sampling with 8 b per colour component and then the DXT compression is performed using 3 threads running in parallel. It should be noted, however, that to decrease latency and computational requirements on the sender, the compressed image is still in YUV colour space with conversion being done on the receiver. For sending with Jumbo frames enabled, the sender machine should have at least 4 cores (as 3 cores are completely busy with compression threads and one remaining core handles the other work). Four cores are also needed in order to send standard-sized 1500 B frames as the load increases only moderately, see Section 3.1.

2.2 Receiver

As noted above, we rely on DXT since its decompression through OpenGL has direct hardware support on a vast majority of modern GPUs. Therefore, an OpenGL front end has been implemented for UltraGrid (the original implementation from ANTLab\textsuperscript{9} used only SDL front end for software-based rendering). After decompressing the image, an OpenGL fragment shader is applied to convert from YUV to RGB colour space. As all image processing occurs on the graphics card, the DXT OpenGL front end uses less resources than any other available front-end for UltraGrid. The OpenGL front end also supports scaling operation, so that the video can be played back using any screen resolution up to 1920×1200.

\textsuperscript{7} http://www.gpgpu.org/
\textsuperscript{8} http://www.evl.uic.edu/cavern/fastdxt/
\textsuperscript{9} https://www.sitola.cz/igrid/index.php/UltraGrid
The implementation has been tested on various NVidia cards (7000 and 8000 series) and it performed as expected. The problems were however when we tried to run the DXT receiver on a Mac Mini with a built-in Intel GMA 950 graphics card as it seems there is no support for OpenGL 2.0 on this GPU (OpenGL 1.4 is only supported). The compilation of shader fragment for colour space conversion fails on this card probably because GLSL extensions are not fully supported—this problem will be worked upon in the future.

3 Performance Evaluation

In order to analyze behaviour of the system, we have performed measurements of end-to-end latencies. This measurement methodology uses the following setup: a generator computer is used to display a time-changing pattern to analyze the latency on an attached LCD screen. The screen of the generator computer is captured by the SONY HVR-Z1E camera attached to the sender computer using an analog to HD-SDI converter (AJA HD10A). The data is received by the receiver computer and displayed on an attached LCD screen. For all the measurements, sender and receiver were connected directly back-to-back (without a switch) using 2 meters of a single-mode fibre. Both generator screen and receiver screen were captured by a still digital camera and the latency difference has been read out. The same principle has been used to carry out end-to-end latency measurements in [2]. This technical report also updates results for uncompressed video shown in [2] as the implementation of both UltraGrid and DVS SDK has improved since then.

The Linux machines (both sender and receiver) for measuring latencies with Centaurus and Centaurus II cards both with and without DXT compression were set up as follows:

- 2× processor AMD Opteron Dual Core 2.6 GHz
- 2 GB RAM
- 10GbE network interface card Myricom Myri-10GE (PCIe, i.e. PCI Express)
- Centaurus (PCI-X) or Centaurus II (PCIe) capture card

The MacOS X machine which has been used again as both sender and receiver has been configured as follows:

- Intel-based Mac Pro
- 2× Intel Xeon Quad Core 3 GHz
- 4 GB RAM
- 10GbE network interface card Myricom Myri-10GE (PCIe)
- Blackmagic Decklink HD Pro capture card (PCIe)
3.1 Results

Latency. Measurement results are summarized in Table 1. Error of the measurement is given by refresh rate of the 60 Hz LCD screen which is 16 ms.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Latency [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux, Centaurus, no compression, Jumbo frames</td>
<td>90±8</td>
</tr>
<tr>
<td>Linux, Centaurus II, no compression, Jumbo frames</td>
<td>85±8</td>
</tr>
<tr>
<td>Linux, Centaurus, DXT compression, Jumbo frames</td>
<td>130±8</td>
</tr>
<tr>
<td>Linux, Centaurus II, DXT compression, Jumbo frames</td>
<td>95±8</td>
</tr>
<tr>
<td>MacOS X, DeckLink Pro HD, no compression, Jumbo frames</td>
<td>148±8</td>
</tr>
<tr>
<td>MacOS X, DeckLink Pro HD, DXT compression, Jumbo frames</td>
<td>178±8</td>
</tr>
</tbody>
</table>

Sender load. Results indicate two positive results: (1) DXT compression has only a very small impact on the latency and (2) latency has improved since publication of results in [2]. Also it is obvious that, in terms of latency, the newer and cheaper Centaurus II card performs better than its predecessor. This is likely due to the fact that it produces a lower computational load on the system, and thus it performs significantly better when DXT compression is occurring (this may be attributed at least in part to its PCIe interface).

CPU load was measured for both Linux and Mac Pro machine senders sending uncompressed and DXT compressed HD video. The results are summarized in Table 2.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>CPU load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux, no compression, Jumbo frames</td>
<td>26 %</td>
</tr>
<tr>
<td>Linux, DXT compression, Jumbo frames</td>
<td>332 %</td>
</tr>
<tr>
<td>Linux, DXT compression, 1500B frames</td>
<td>350 %</td>
</tr>
<tr>
<td>MacOS X, no compression, Jumbo frames</td>
<td>46 %</td>
</tr>
<tr>
<td>MacOS X, DXT compression, Jumbo frames</td>
<td>324 %</td>
</tr>
<tr>
<td>MacOS X, DXT compression, 1500B frames</td>
<td>350 %</td>
</tr>
</tbody>
</table>

The CPU load is measured in total for all CPU cores. 100% CPU load in this case means either one fully loaded core or equivalent load distributed among a number of cores. In the case of sending DXT compressed streams the DXT compression was fully loading three cores on both Linux and Mac Pro sender. Such a requirement was easily matched with quad-core (Linux sender) and octo-core (Mac Pro sender) setups.
Receiver load. CPU loads on the receiving machines and different HD video streams are given in Table 3. An expected observation is a low CPU load imposed by receiving and displaying DXT compressed streams. For the Linux receiver, the CPU load was only 17% of an Opteron 245 core when using DXT compressed stream with Jumbo frames enabled.

Table 3. Receiver CPU load evaluation results summary.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>CPU load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux, no compression, Jumbo frames</td>
<td>123 %</td>
</tr>
<tr>
<td>Linux, DXT compression, Jumbo frames</td>
<td>17 %</td>
</tr>
<tr>
<td>Linux, DXT compression, 1500B frames</td>
<td>24 %</td>
</tr>
<tr>
<td>MacOS X, no compression, Jumbo frames</td>
<td>160 %</td>
</tr>
<tr>
<td>MacOS X, DXT compression, Jumbo frames</td>
<td>9 %</td>
</tr>
<tr>
<td>MacOS X, DXT compression, 1500B frames</td>
<td>29 %</td>
</tr>
</tbody>
</table>

Traffic profiles. UltraGrid is known for generating bursty traffic in order to deliver video frames to the destination as fast as possible [2]. We have compared traffic profiles for DXT compressed video with both standard sized and Jumbo frames and uncompressed video with Jumbo frames. The profiles were captured using the `tcpdump` utility on the Mac Pro sender machine. Only the first 68 B of each packet were captured in order to minimize the performance impact of the measurement itself.

The results are shown in Figures 3, 4, and 5. The raw packet arrival times were aggregated into 1 ms time slots, which are used to plot the profiles. Most extreme bursts in terms of packet count are generated when transmitting DXT compressed video with standard-sized frames. Both compressed and uncompressed streams with Jumbo frames generate similar packet rates, but for compressed video the burst length is significantly shorter.

4 Conclusions

In this report, we described the DXT compression implementation for UltraGrid, a low-latency high-definition collaborative system. The OpenGL front end for UltraGrid has been implemented, and more importantly, this allowed for real-time image compression utilizing a modified version of FastDXT library and decompression using the OpenGL front end. This compressed version of UltraGrid required only 250 Mbps of bandwidth, 20% of CPU utilization during data receiving on an Opteron 245, and provided quite usable quality and stability. Although the compression process is computationally intensive, it is a highly parallel task,
and results in minimal latency impact when compared to other compression techniques. The end-to-end latency of the uncompressed HD video was significantly lower compared to the results reported in [2], which can be attributed to optimization of the software since 2005, and utilization of the new DVS SDK.

The whole system was demonstrated at the Global Lambda Infrastructure Forum (GLIF) in Prague (September 2007) as part of the CoUniverse demonstration organized by Masaryk University and during the Center for Computation and Technology demo at Super Computing 2007 (November 2007).

As for future work, we would like to focus more on real-time compression techniques using GPUs, namely on the new generation of NVidia GPUs that will allow us to use the CUDA programming model. As those newer cards are designed with general purpose calculations in mind, we expect much better performance in terms of read-back latency.
DXT compressed video, 1500B frames, 10GbE

DXT compressed video, 8500B frames, GbE

**Figure 4.** Traffic profiles for various UltraGrid setups.
DXT compressed video, 8500B frames, 10GbE

Uncompressed video, 8500B frames, 10GbE

Figure 5. Traffic profiles for various UltraGrid setups.
References


12 http://oss.sgi.com/projects/ogl-sample/registry/EXT/texture_compression_s3tc.txt
CAVE to CAVE: Communication in a Distributed Virtual Environment

Roman Berka, Zdeněk Trávníček, Vlastimil Havran, Jiří Bittner, Jiří Žára, Pavel Slavík, Jiří Navrátil

Abstract

A virtual environment (VE) is often used as a tool in the design process or as a part of visualization techniques which make it possible for the user to enter a space which is otherwise not accessible or does not even exist. Virtual environments can also be used for collaboration when some of the users are not present locally. Remote users communicate with local users via a network which requires a configuration based on more VEs (e.g. CAVE-like devices) connected together. The applications used to control such devices are usually designed to communicate with the same type of remotely connected system. On the other hand, communication between heterogeneous systems becomes increasingly important with the technological advances in presentation technologies. This report describes a project which aims to transmit video signals between heterogeneous VE devices over a high speed network connection.

Keywords: virtual environment, CAVE, high-speed network

1 Motivation

1.1 A Case Study

The VEs based on CAVE-like devices [3], [4] are usually used to visualize any modelled virtual environment (e.g. evaluation of a new interior design) or is not accessible (e.g. the surface of a distant place, a planet, the bottom of the ocean). These devices can be entered simultaneously by more than one user. A configuration of the CAVE system with three projection walls is shown in Figure 1. It usually uses software producing panoramic stereoscopic views to the virtual scene in real-time. Using equipment to track motions of users it is possible to interact with the presented scene and its parts. Two or more CAVE systems can be connected together and the immersion of the visualized data can be shared by different groups of users. Each of the groups can stay in their CAVE laboratory and just share the virtual environment using the network connection. This allows for example to perform shared walk-throughs of the VE or common exploration of visualized simulation processes.

The remote connection of VEs can be divided into several categories based on the degree of interactivity between its participants:
1. Remote presentation
2. Remote interaction
3. Interaction in a collaborative virtual environment

1.1.1 Remote presentation

Remote presentation makes it possible to passively enter the remote virtual space and watch processes inside. As an example we can imagine virtual remote visits of groups to historical buildings or to “Jurassic Park”-like places. The difference from a normal movie is that in this case all (or most) presented material is computed in real-time. This brings a new quality which could be compared with the idea of “Kinoautomat” from the 1960’s, but owing to contemporary technologies we can reach a new form between classical movie and theatre. The processes inside the scene can be then influenced by various events changing actual conditions (e.g. weather simulations or by branching the story through observers’ voting).

The technical solution using transport of the multimedia data over the network allows a parallel presentation to more groups of observers located at different places around the world. The scheme of such an application based on equipment developed by our team is presented in Figure 2. Here, the visualizations are computed by the computer cluster of the CAVE system and are presented to the users inside the CAVE where an interaction with the scene is also possible. In addition, the visualized output is also transmitted to the remote stereoscopic wall, where another group of users can passively watch the processes and the interactions taking place in the CAVE.
1.1.2 Remote interaction

The remote interactive application allows the user to walk through a remote virtual scene and to interact with objects inside. The virtual environment can represent a real scene built from pre-measured data. This configuration expects that the users are equipped by a tracking system which allows interaction with the scene. There are several applications where a group of users can share a scene with another distant group [6]. In addition, motion tracking gives a possibility of interaction. Such a typical application then enables different groups of users to edit some common data (architecture, scientific visualizations, etc.).

Figure 2. Passive remote presentation scheme CAVE stereo wall

Figure 3. Interactive remote presentation CAVE stereo wall

The remote interaction can use various resources to influence the input data,
e.g. the remote control of a mobile robot which is able to obtain geometric representation of the scene. Then it is possible that the user can walk through a virtual scene sent by the remotely controlled robot (e.g. underwater scenery, or a dangerous environment; such as house on fire).

The scene representations sent by a distant source of data can be then rendered by the CAVE computer cluster and distributed to other VE devices. Owing to possibility of interaction all participating groups of users can remotely control the process (e.g. motions of the robot).

1.1.3 Interaction in a Collaborative Environment

In this case, the rendered scene and the real video of users are combined together with the help of motion tracked data. This allows the users to see not only the shared scene and interact with it but they can also see their remote colleagues in the form of a video rendered and mixed directly in the virtual environment [1].

![Figure 4. Mixed reality CAVE stereo wall](image)

The technical concept is then based on a similar idea like TV virtual studio where live video is composed by keying virtual scenes. The application of such an approach then leads to a distributed collaborative environment based on the video signal distribution and on real-time composition.

1.2 Goal of the Project

To connect the CAVE-like devices we need a protocol which enables the communication and interaction of all participating users. There already exists the possibility of communication between two CAVE-like systems based on transmission
CAVE to CAVE: Communication in a Distributed Virtual Environment

of vector-data representing changes in the virtual world. This approach usually expects a homogeneous distributed environment with the same clients on both sites. The implementation of this approach often use the CAVElib library, which also have to be used by the distant client so that it can receive the transmitted data.

In contrast we propose to use a standard data format for communication, which makes the presented data more accessible for remote users using standard visualization tools as, for example, a common video-player. Specifically, we propose to grab and stream a real-time stereo video signal that is used to share the views to the VE.

The goal of the project is to develop methods, based on contemporary technologies, enabling VE to be shared by users independently on various VR devices and making it possible to collaborate inside the VE. To satisfy this goal we will try to find a way how to take the necessary data from a VE device computer controlling system and transmit them to the remote system. The first approach will be based on video data transmission and the remote system is not necessarily the same type as the local one.

2 The Project Progress in 2008

The project CAVE2CAVE (C2C) is an activity of three cooperating partners: Institute of Intermedia, Faculty of Electrical Engineering, Czech Technical University in Prague (IIM), Department of Computer Graphics and Interaction, Faculty of Electrical Engineering, Czech Technical University in Prague (VRLab), and CESNET, association of legal entities. The first two partners have laboratories working in the area of computer graphics, multimedia and virtual reality, CESNET plays the role of the connectivity provider. The goals for the year 2008 were characterized by the preparation of equipment and concepts which were then evaluated. First of all, the method of graphical information acquisition from virtual devices has been designed and tested (see the following text). Each of the three cooperating partners had to set up the technology and equipment associated with the particular role of the partner in the project.

2.1 Institute of Intermedia

IIM, a laboratory equipped with a virtual reality system - CAVE, prepared the device for connection to the high bandwidth network technology and prepared the implementation of the video grabbing module. The development and testing process of this method is described further in this section.

IIM also initiated the preparation of tools enabling acquisition of a stereo-
video signal using a special stereo-camera system. This tool was prepared as a part of a student project and it will be used in the next phase of the C2C project during 2009. This constructed system represents a stereo video production chain which can be used to capture real video and transport it to the remote projection system. The configuration of the system is based on a two-fixed-camera system (Figure 5) connected to the controlling computer by two ISO1394 cables or directly connected to a network using an ethernet interface (Figure 6). The control computer then processes two video signals and sends them to the remote presentation system. The two-channel video then can be presented on stereoscopic projection systems or an auto-stereoscopic system.

Two industry-level cameras with 1Gbit Ethernet interface and auto-stereoscopic display were acquired during 2008 and will be incorporated in the project equipment in the next project phase. Another tool prepared during students’ participation in the project is the motion tracking system VICON which was tested for use in the CAVE system to track the user and his interactions. Both the presented tools are parts of the distributed VR collaboration system described above in Remote Interaction and Interactive Collaborative Environment case studies.
2.2 Laboratory of Virtual Reality (VRLab), DCGI

The second side of the communication chain has been prepared in the VRLab. The most important was to set up the stereoscopic wall (supplied by EON Reality) and to acquire computer equipment, consisting of two control computers connected directly to IIM using an optical 10Gbps link and the computational server. Next, the VRLab also tested the motion tracking system (A.R.T.) newly acquired in 2008. Using all these devices with the already mentioned special 1Gb cameras will enable key experiments with a collaborative distributed environment planned for 2009.

2.3 CESNET

The role of CESNET was to set up the connection between IIM and VRLab using a 10Gbps optical link. This was successfully realized and the link has been tested in the third quarter of 2008. In addition, CESNET supported acquisition of the necessary technical devices (control computers for the stereoscopic wall in VRLab, special 1Gb cameras, and auto-stereoscopic display as an alternative to the stereoscopic projection systems).
3 Technical Details of Problems Solved in 2008

3.1 Connecting VR Devices

In many applications more VR devices communicate in order to establish a collaborative environment [2]. Here, the vector-data and video signals are used allowing users to exchange audio-visual information. The communication of such devices requires a suitable protocol. As there exist more solutions of how to acquire the data from the source, we have to recall the whole visualization pipeline. Figure 7 represents the pipeline where data describing a scene are loaded by an application which stores them in its internal form. The data are then sent to a GPU, rendered and sent to the output in the form of a VGA/DVI signal.

The first possibility lies in a simple data exchange at the level of the shared file system. This approach expects that local and remote applications have access to the same file and requires synchronization of the appropriate processes.

The second approach is based on communication at the level of the vector-data exchange between applications [7]. This is the most typical solution often existing in the form of a run-time library. One example could also be the library CAVElib which is used by many CAVE systems around the world, including the one at the Institute of Intermedia at the CTU in Prague.

The next possibility is based on the idea of forking the flow of OpenGL commands before they are sent to the GPU and transmit them also to the remote station where these commands are then executed. This solution requires a remote cluster with enough power to perform the rendering process and some modifications in the remote execution pipeline, for example remapping of IDs for display lists. On the other hand this method also has advantages. One of them is the possibility of simple conversion from monocopic channel to stereoscopic reproduction on the remote system.

![Data flow scheme](image)

**Figure 7.** Data flow scheme

The fourth method is based on a process of reading rendered data directly
from the GPU buffers and sending them to the remote VR system. This case is related to the video-signal transmission. The weakness of this method lies in a bottleneck represented by the process reading the video-buffer. We will show our solution of this problem next in this report.

Finally, the fifth method uses a hardware solution when the video-signal is captured directly from the VGA/DVI output of the graphical subsystem converted to video-stream and then sent through the network. This approach is also based on the video/signal transmission. Its advantage is the rate of data processed but the disadvantage could be the impossibility of controlling the render process. There is just one resolution and frame rate on a graphic card output. Thus this approach has a lower degree of usability when synchronization control of the multiple channel signal (e.g. stereo video) is required.

![Figure 8. Direct frame buffer access method](image)

4 The Video Transmission Based Methods

The communication of VR devices using video signal transmission is not a typical problem. We require a fast and controllable data source and high bandwidth according to resolution and frame-rate. On the other hand, video signal transmission allows the use of video transmission standards (e.g. for streaming) and this allows for connection of VR devices independently of the system of remote devices. In addition, many video-processing algorithms can be used during the transmission.

As the main problem of video-based communication was to get the signal from the graphics card, we tested three solutions how to achieve it. The first approach is naive and its idea is to directly read the content of the video-buffer. The second approach is based on OpenGL command transmission and the last approach refines the first one using Pixel Buffer Object extension of OpenGL.
4.1 Direct Real-Time Video Buffer Access

The direct video-buffer access method is presented here by scheme in Figure 8. The figure shows the pipeline consisting of the rendering block, the grabbing block and the swap buffer call. After the frame N is rendered, the content of the video memory buffer actually filled is read and the data are sent to the next processing stage to be transmitted. The grabbed data are usually processed by another thread. Then the glxSwapBuffers() function is called and frame \(N + 1\) can be rendered.

This method conceptually meets our requirements because we have full control over the image quality and synchronization. On the other hand the method does not work successfully in a real situation due to low bandwidth between the graphics card and the main memory. We used this method just to test our concept.

4.2 OpenGL Command Transmission

The OpenGL command transmission method is based on the interesting idea of sending just commands normally sent to the GPU through OpenGL API. Each OpenGL call is also sent to the remote system which executes it after necessary modifications. This method does not correspond to our original goal and thus we partially left it as a minor task.

4.3 Parallel Video Buffer Access

The parallel video buffer access method is based on the use of an OpenGL extension called Pixel Buffer Object (PBO), see Figure 9. Here, after the frame \(N - 1\) is rendered the wrapper of function glxSwapBuffers() initiates the grabbing process of this frame. This does not block the next operations and the original function glxSwapBuffers() is called and the next frame \(N\) can be rendered. As the rendering process is controlled by the GPU and the grabbing process by the DMA, both can be running simultaneously. The data of frame \(N - 1\) are transferred to the main memory while the frame \(N\) is rendered to the second buffer.

This method we finally tested as the core of our video based communication between two VR systems. The implementation based on the use of PBO extension works as an OpenGL wrapper which is applicable on any OpenGL program including applications implemented using the SDL library. The content of the graphics window is sent to the remote system where it can be presented. Such a solution has relatively low computational requirements.

The data are transported through the standard RTSP/RTP protocol as an uncompressed video meeting standard RFC 4175 [5]. The implementation is build
on libraries live555\textsuperscript{1} and libavutil (part of FFmpeg\textsuperscript{2} project) under the Linux operating system. The method has been tested in an OpenGL application running on local system with four different resolutions simulating the different data bit-rate requirements. The real-time computation produced a sequence of images which has been sent to the network using the RTP protocol.

The configuration of the laboratory experiment is illustrated by the scheme in Figure 10. One computer node of our CAVE cluster has been used as a source of testing video signal. The rendered data captured from the graphics card have been scaled using libswscale library, which is also part of the FFmpeg\textsuperscript{3} project, and encoded by our proprietary application implemented along the standard RFC 4175 [5]. Next, the resulting data have been sent to a client computer as a video stream (using live555 library) via a dedicated 1Gb network connection containing one 1Gb switch. Finally, the video stream received on the client side has been processed in reverse order to give the final video presentation.

The complete transmission chain with the basic functionality of the video capturing wrapper, transmission mechanism using standard protocols and simple presentation component gave us a proof of concept which is to be developed during the next project phases.

\textsuperscript{1} http://www.live555.com/
\textsuperscript{2} http://www.ffmpeg.org/
\textsuperscript{3} http://www.ffmpeg.org/
5 Features and Results

The method of parallel video buffer access has been tested on various data bandwidth demands. The measured results are presented in Table 1.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01 (100×100)</td>
<td>18.3</td>
<td>21</td>
<td>lossless transmission</td>
</tr>
<tr>
<td>0.09 (300×300)</td>
<td>164.8</td>
<td>174</td>
<td>lossless transmission</td>
</tr>
<tr>
<td>0.3 (640×480)</td>
<td>562.5</td>
<td>590</td>
<td>small data loss</td>
</tr>
<tr>
<td>0.48 (800×600)</td>
<td>879.9</td>
<td>810</td>
<td>significant data loss</td>
</tr>
</tbody>
</table>

The testing video signal was transmitted in the RGBA format (32bpp) and with a frame-rate of 60Hz which corresponds to two streams carrying two video channels each with a frame-rate of 25-30Hz.

Evaluating the presented numbers we can summarize them so that resolutions 800×600 and higher, which are typical for VR systems used in our equipment, generate bit-rates over 1Gbps per video-channel. The case of the 6 video channels, which corresponds to a 3-wall CAVE system, represents bit-rates requiring connection with 10Gbps capacity.

The results also show that our parallel video buffer access method is able to produce a data rate which is enough for the further scheduled project experiments. This method also offers several interesting features.

Using the video stream based communication between VE devices we can remotely present the VE to a distant user which is not expected to use the same device as the local one. In addition, the stream can be presented on more than one remote device using multicast communication.
This idea extends our case studies to situations when interactions between two virtual systems can be observed by more users using standard computer equipment.

6 Future Work

The near future of the project will be dedicated to activities leading not only to stereo video transmission but also exchanges of other types of data including vector descriptions of 3D objects and tracking. The tracked data are typically used when interacting with objects inside the virtual scene. Exchanging video data mixed with other types of data we can extend the form of distributed VE to a more realistic collaborative tool profiting from wide bandwidth networks.

7 Conclusion

Based on the proposed goals we can summarize the achieved results as the first successful stage where the core technique has been implemented and evaluated. This means that we have a basic tool which can now be incorporated into a set of applications corresponding to our case studies presented above. The technique implemented and presented here does not represent any new technology per se, but it represents a new approach in the context of contemporary technologies and the way of their application.

References


---

CESNET Association and CESNET2 Network

CESNET is an association of all government-founded universities of the Czech Republic and the Czech Academy of Sciences. Its main goals are:

- operation and development of the Czech NREN (National Research and Education Network),
- research and development of advanced network technologies and applications,
- broadening of the public knowledge about the advanced networking topics.

CESNET is a long-time Czech academic network operator and participant of corresponding international projects. The most important international relationships of the CESNET association are:

- shareholder of DANTE (*Delivery of Advanced Network Technology to Europe*),
- member of TERENA (*Trans-European Research and Education Networking Association*),
- Internet2 international partner since 1999,
- Partner of the European GN3 project.
- founding member of GLIF (*Global Lambda Integrated Facility*)

More information about CESNET may be obtained from [http://www.ces.net](http://www.ces.net).

**CESNET2 topology as of May 2009**

![CESNET2 topology map](image)