



OPTIMIX: OPTIMISATION OF MULTIMEDIA OVER WIRELESS IP LINKS VIA X-LAYER DESIGN

DELIVERABLE D3.1C

REFINEMENT OF SPECIFICATION, FINAL DESIGN AND ANALYSIS OF TRANSPORT AND NETWORK LAYER PROTOCOLS AND MECHANISMS

| | |
|------------------------------------|------------------|
| Contractual date of delivery to EC | M22 (31/08/2010) |
| Actual date of delivery to EC | 31/08/2010 |
| Revision date (if applicable) | |
| Version number | 1.0 |

| | |
|--------------|-------------------|
| Editor(s) | BME |
| Participants | CEFRIEL, VTT, BME |

| | |
|--|-------------------|
| Deliverable nature and security level | Report, PU |
| Estimated work-load | |
| Total number of pages | 45 |



Executive Summary

This deliverable is the third in the series, which deals with transport and network layer and their functionalities in the OPTIMIX communication chain. As has been already investigated in the previous deliverables D3.1a and D3.1b, the IP QoS strategies plays an important role in transporting multimedia packets from the source to the sinks. This deliverable is partly devoted to some more IP level QoS strategies.

This document is the third reporting document of Task 3.1. For this reason, (compared to the first two deliverable which provided a description of possible technologies), the main goal of this document is to provide the final results connected with the considered technologies. Among other things, the reader is informed about the results of multicast support in the OPTIMIX scenario, e.g. the application of DCCP transport protocol for multicast transmission, QoS support in IP networks and security mechanisms to provide privacy of the multicast flows. For a complete view on the network and transport layer related issues the reader is referred to the D3.2c “Final design and functional/performance analysis of network architecture” document, which contains some related, but non-overlapping areas.

TABLE OF CONTENTS

| | |
|---|-----------|
| EXECUTIVE SUMMARY | 2 |
| 1 INTRODUCTION | 5 |
| 2 MULTICAST SUPPORT..... | 6 |
| 2.1 DETAILED OPERATION | 6 |
| 2.2 SIMULATION ANALYSIS..... | 6 |
| 2.3 DCCP FOR MULTICAST..... | 7 |
| 2.3.1 <i>Related Work</i> | 8 |
| 2.3.2 <i>Problems and Solutions of DCCP multicast</i> | 8 |
| 2.3.3 <i>Proposed DCCP multicast model</i> | 10 |
| 2.3.4 <i>Benefits and Enhancement of DCCP multicast</i> | 11 |
| 2.4 DCCP FEEDBACK MANAGEMENT | 12 |
| 2.4.1 <i>Selecting client at connection setup</i> | 13 |
| 2.4.2 <i>Change client during the communication</i> | 14 |
| 2.4.3 <i>Selected client leaves the multicast group</i> | 15 |
| 2.4.4 <i>New client joins the multicast group</i> | 15 |
| 3 QOS OVER IP NETWORKS..... | 16 |
| 3.1 ABSOLUTE QOS IN IP NETWORK..... | 16 |
| 3.1.1 <i>Absolute QoS in a IP network: promoting and downgrading UDP traffic</i> | 16 |
| 3.1.2 <i>Detailed analysis of AWTP to achieve absolute QoS</i> | 25 |
| 3.2 IMPACT OF THE DESIGNED TRAFFIC ENGINEERING MECHANISM WITH FAULT RESILIENCE CAPABILITY | 34 |
| 4 SECURITY..... | 37 |
| 4.1 SRTP MODULE | 37 |
| 4.2 PERFORMANCE OF THE CHACHA CIPHER | 37 |
| 4.3 ENHANCED ASPACE ALGORITHM | 38 |
| 4.3.1 <i>Counter block</i> | 38 |
| 4.3.2 <i>Counter settings</i> | 38 |
| 4.3.3 <i>MAC difference calculation</i> | 39 |
| 4.4 PARTIAL INTEGRITY..... | 40 |
| 5 CONCLUSION | 43 |
| 5.1 REFERENCES..... | 44 |
| 5.2 GLOSSARY..... | 45 |

LIST OF TABLES

| | |
|--|----|
| Table 1 – Example of receive rates of the clients (sending rate is 500kbps) | 14 |
| Table 2 – Another example of receive rates of the clients (sending rate is 500kbps)..... | 15 |
| Table 3 – first set of configuration parameters | 19 |
| Table 4 – Transmission strategy for 6m | 41 |
| Table 5 – Transmission strategy for 2 rooms..... | 42 |

LIST OF FIGURES

| | |
|--|----|
| Figure 1 – Unicast vs. multicast transmission of the same content to three terminal equipments. | 6 |
| Figure 2 – Visualization of reception for the chosen terminals (green: receiving 0 stream, blue: receiving 1 stream, magenta: receiving 2 streams, red: receiving 3 streams). | 7 |
| Figure 3 – Initiation, Data transfer, Termination with states | 8 |
| Figure 4 – Client, server states..... | 8 |
| Figure 5 – DCCP synchronization | 9 |
| Figure 6 – Valid Sequence numbers | 9 |
| Figure 7 – Acknowledgement Number validity..... | 9 |
| Figure 8 – Multiple DCCP-Acks | 10 |
| Figure 9 – DCCP multicast scheme | 11 |
| Figure 10 – DCCP-Ack aggregation..... | 12 |
| Figure 11 – DCCP packet sequence chart..... | 13 |

| | |
|--|----|
| Figure 12 - Network scenario | 19 |
| Figure 13 - Promotion by threshold with $K_{up}=5$, downgrading by threshold with $K_{down}=20$ | 20 |
| Figure 14 - Promotion by moving average with $K_{up}=5$, downgrading by threshold with $K_{down}=20$ | 20 |
| Figure 15 - Promotion by low pass filter with $P_{up}=0.9$, downgrading by threshold with $K_{down}=20$ | 21 |
| Figure 16 - Promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$ | 21 |
| Figure 17 - Promotion by threshold with $K_{up}=2$, downgrading by threshold and $K_{down}=20$ | 21 |
| Figure 18 - Promotion by low pass filter, downgrading by threshold with $K_{down}=10$ | 22 |
| Figure 19 - Promotion by low pass filter, downgrading by moving average with $K_{down}=5$ | 22 |
| Figure 20 - Promotion by low pass filter, downgrading by moving average with $K_{down}=10$ | 23 |
| Figure 21 - Promotion with low pass filter, downgrading by moving average with $K_{down}=20$ | 23 |
| Figure 22 - Promotion by low pass filter, downgrading by low pass filter with $P_{down}=0.99$ | 23 |
| Figure 23 - Promotion by low pass filter, downgrading by low pass filter with $P_{down}=0.8$ | 23 |
| Figure 24 - AWTP, promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$ | 24 |
| Figure 25 - DWFQ, promotion by low pass filter with pole $P_{up}=0.9$, downgrading by threshold and $K_{down}=20$ | 24 |
| Figure 26 - AWTP, 99th percentile of the end-to-end delay of the UDP traffic: it starts in the first class (blue) than is promoted to the second class (red), to the third class (green) and finally to the best class (cyan). The threshold for promotion is fixed to 10 msec | 25 |
| Figure 27 - DWFQ, 99th percentile of the end-to-end delay of the UDP traffic: it starts in the first class (blue) than is promoted to the second class (red), to the third class (green) and finally to the best class (cyan). The threshold for promotion is fixed to 10 msec | 25 |
| Figure 28 - OPTIMIX network architecture | 26 |
| Figure 29 - AWTP, promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$, feedback delay of 20 msec. | 27 |
| Figure 30 - AWTP, 99th percentile of the end-to-end delay of the UDP traffic with a feedback delay of 20 msec | 27 |
| Figure 31 - AWTP, promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$, feedback delay of 50 msec. | 28 |
| Figure 32 - AWTP, promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$, feedback delay of 1000 msec. | 28 |
| Figure 33 - AWTP, 99th percentile of the end-to-end delay of the UDP traffic with a feedback delay of 50 msec | 28 |
| Figure 34 - AWTP, 99th percentile of the end-to-end delay of the UDP traffic with a feedback delay of 100 msec | 29 |
| Figure 35 - End-to-end delay of the UDP traffic that can change class, in the case of 20 msec of feedback delay | 29 |
| Figure 36 - End-to-end delay of the UDP traffic that can change class, in case of 50 msec of feedback delay. | 30 |
| Figure 37 - End-to-end delay of the UDP traffic that can change class, in case of 100 msec of feedback delay. | 30 |
| Figure 38 - Scenario with 6 nodes | 30 |
| Figure 39 - AWTP, promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$, feedback delay of 20 msec, and 6 nodes | 31 |
| Figure 40 - AWTP, 99th percentile of the end-to-end delay of the UDP traffic with a feedback delay of 20 msec and 6 nodes | 31 |
| Figure 41 - End-to-end delay of the UDP traffic that can change class, in the case of 20 msec of feedback delay and 6 nodes | 32 |
| Figure 42 - AWTP, promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$, feedback delay of 20 msec and quality factors 1, 4, 8, 16 | 33 |
| Figure 43 - AWTP, 99th percentile of the end-to-end delay of the UDP traffic with a feedback delay of 20 msec and quality parameters 1, 4, 8, 16 | 33 |
| Figure 44 - End-to-end delay of the UDP traffic that can change class, in case of 20 msec of feedback delay and quality parameters 1, 4, 8, 16 | 34 |
| Figure 45 - Cipher throughputs | 38 |
| Figure 46 - Enhanced APACE algorithm | 39 |
| Figure 47 - The old version of the APACE algorithm | 39 |
| Figure 48 - Linear and non linear calculations | 40 |
| Figure 49 - Transmission in the 6m scenario | 41 |
| Figure 50 - Transmission in the 2 rooms scenario | 41 |

1 Introduction

This document reports the results of the research conducted in the area of network and transport protocols. The content presented here relies on the first two deliverables of Task 3.1, namely D3.1a [6] and D3.1b [7], which serve as introduction to this deliverable. The reader should expect final results in this document, since Task 3.1 closes with this final research report D3.1c: the information stored here serves as a final report on the activities carried out in Task 3.1 of the OPTIMIX project.

It should be mentioned that this document (D3.1c) should be read together with D3.2c “Final design and functional/performance analysis of network architecture”. The latter deliverable is more about network transparency issues; however, there are areas in which distinctions are difficult to make. For instance, the reader will find the description of HIP based mobility related simulation results in D3.2c, although the HIP is located between the network and transport layers.

This document provides some results in the simulation and research of transport protocols. Traditional protocols (e.g. UDP and TCP) are not considered here, since they have a huge amount of drawbacks compared to more recent transport technologies (like DCCP). The OPTIMIX project wants to use DCCP in the demonstration and simulations, which is more advanced protocol compared to traditional UDP and TCP. The primary characteristics of DCCP and SCTP have been reported in D3.1b [7]. There, the basic properties of these transport protocols have been thoroughly analyzed. Following the decision of the OPTIMIX partners, DCCP will be used as a transport protocol. However, DCCP has been created as a unicast supporting protocol [2]. The reader can find here the details of the investigations about the use of DCCP in a multicast environment. All required modifications of the basic DCCP protocol are described in this document, together with details on the implementation.

Considering the network characteristics, the most important metric is the quality of service (QoS) from the user point of view. At the network level, routers must take care of different priorities of packets to support an overall QoS framework for all the clients. The final results of the simulation of the methods to support QoS are given here.

This document is organized as follows: in Section 2 the reader will find the description of multicast support. After a brief introduction of the multicast operation itself in Section 2.1, some simulation analysis follows in Section 2.2. The extension of DCCP transport protocol to the multicast case is described in Section 2.3. Section 2.4 deals with the feedback handling, when multiple receivers gets the multicast stream with DCCP transport protocol. Section 3 is about the quality of service issues in the IP network. The method of supporting absolute QoS is described first in Section 3.1. Next, Section 3.2 is about the final results of traffic engineering. Section 4 is about the security related research results, Section 4.1 details the SRTP module’s operation, Section 4.2 is about the ChaCha cipher, Section 4.3 is about the enhanced APACE algorithm, and finally Section 4.4 details the partial integrity issues.

2 Multicast support

For proper and economical operation of a streaming system like OPTIMIX it is indispensable to use network resources as optimally as possible. A common solution for traffic reduction in the backbone network involves the utilization of multicast solutions to serve a vast number of end-users with the same (usually live) content. In a multicast scenario, the content is streamed simultaneously to many users forming multicast groups, and this is done without the unnecessary duplication of transport packets in the access network. The concept is depicted in Figure 1. Note that no duplication of packets occurs in the multicast case; hence the burden on the access network is decreased considerably.

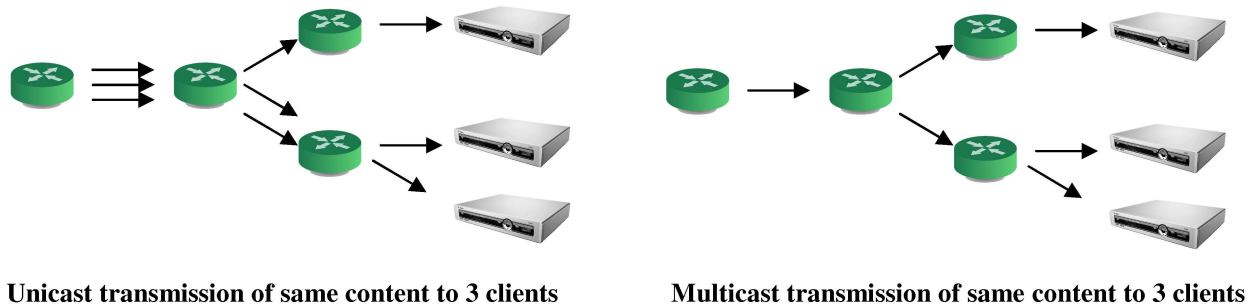


Figure 1 – Unicast vs. multicast transmission of the same content to three terminal equipments.

In this section, we present the implementation of multicast support to the common OPTIMIX simulation chain.

2.1 Detailed operation

In the OPTIMIX simulation environment, multicast streaming is implemented in the traditional way: multicast groups with a predefined multicast address and port are formed for each piece of multimedia content. Multimedia content can consist of an AVC stream and optionally a corresponding AMR audio stream; an SVC base layer and a corresponding AMR audio stream; or an SVC enhancement layer.

The terminal equipments representing the end users can enrol for and leave a multicast group by sending Multicast Listener Discovery (MLDv2) State Change Record messages encapsulated in ICMPv6 packets. State Change Record messages might be of record type `ALLOW_NEW_SOURCES` for group join and of record type `BLOCK_OLD_SOURCES` for group leave requests. Currently, these are the only IPv6 multicast message types supported in OPTIMIX. Moreover, the addressing scheme within these messages is slightly different from what is described in the standard [9th]. Nevertheless, this difference has no influence on functionality.

In case of SVC streams, the terminals might enrol for just the SVC base layer (with or without audio) or the SVC base layer and one or more enhancement layers. When the content is no longer needed by the terminal equipment, or if due to link conditions, it is necessary to decrease the bandwidth, another MLDv2 message is generated and sent up to the network.

MLDv2 State Change Record messages are propagated upstream in the network through the base station MAC layer (BS MAC) to the server IP layer. Once an MLDv2 `ALLOW_NEW_SOURCES` message from terminal equipment (TE) reaches the BS MAC, the BS MAC checks in its neighbour table (a table that contains the MAC address of the connecting terminals) whether any connecting terminal equipments are enrolled to the newly requested content. If so, it adds an entry with the requesting TE to the neighbour table and discards the `ALLOW_NEW_SOURCES` message. If no other connecting terminal equipment is receiving the requested stream, the BS MAC adds the appropriate entry to its routing table and forwards the message to the core network IP routing module (IP ROUTER). For any incoming `ALLOW_NEW_SOURCES` message, the IP ROUTER checks whether a route exists to the appropriate terminal. If it exists, the message is discarded; otherwise, a routing table entry is created. As soon as the table entry is added to the routing table, the multicast packets are forwarded first to the base station the requesting terminal is connected to, then to the terminal itself. Thus, multicast routing occurs in layer 2 in the base station, and in layer 3 in the core network.

Processing of MLDv2 `BLOCK_OLD_SOURCES` messages is implemented in a similar fashion. As soon as the BS MAC receives an MLDv2 `BLOCK_OLD_SOURCES` message from one of its terminals, it checks whether other connecting terminals are receiving the content corresponding to the multicast address in the message. If so, it deletes the corresponding entry in the neighbour table and discards the message. If, on the other hand, no other connecting terminals are entitled for the content, the BS MAC forwards the MLDv2 `BLOCK_OLD_SOURCES` message to the IP ROUTER and deletes the neighbour table entry. The IP ROUTER acts similarly; it checks whether a route exists from the server to terminal for the multicast content, then it deletes the appropriate route.

2.2 Simulation analysis

Evaluation of the multicast mechanism was carried out on a “pass-fail” basis. Five terminals were selected for receiving SVC multicast content. Two of the four terminals were enrolled for SVC base layer and one enhancement layer, one for only the base layer and one for the base layer and two enhancement layers. One terminal was not receiving any streams.

The time stamps when each terminal joined and left a multicast group were prescribed in a settings file, and the reception of multicast content was visualized by assigning different colours for the terminals according to the number of received multicast streams (Figure 2).

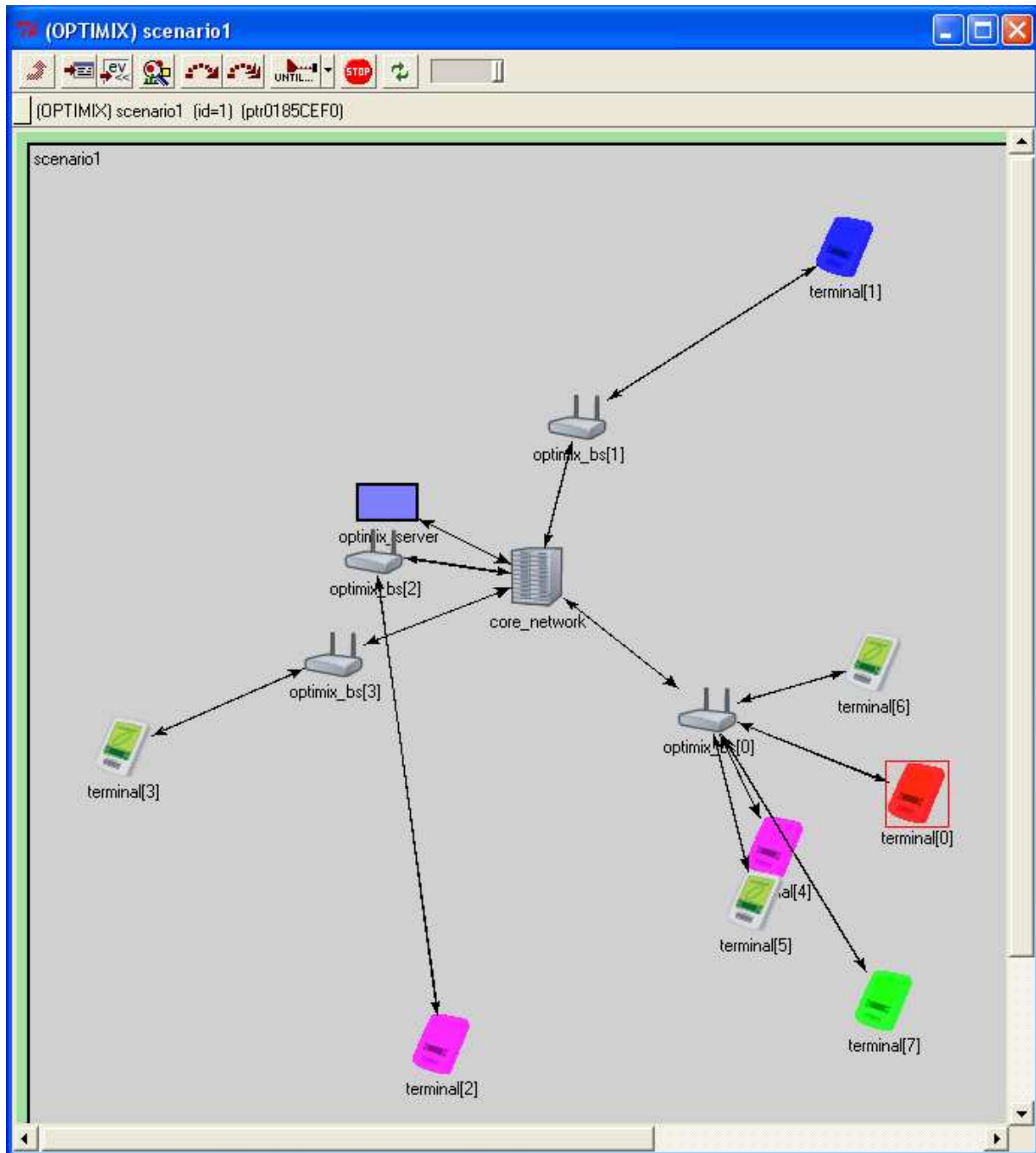


Figure 2 – Visualization of reception for the chosen terminals (green: receiving 0 stream, blue: receiving 1 stream, magenta: receiving 2 streams, red: receiving 3 streams).

The time of arrival of the multicast packets together with the SVC layer ID was also recorded. All conducted tests resulted in a pass.

2.3 DCCP for multicast

DCCP [2] is a unicast, connection-oriented protocol with bidirectional data flow. Unfortunately none of the main DCCP mechanisms, be it connection setup, acknowledgements, or even congestion control, apply naturally to multicast. We have collected the arising problems and tried to find solutions to make the DCCP protocol capable for multicast data delivery.

2.3.1 Related Work

Several approaches were studied where the reliable TCP was extended to support multicast. Making TCP capable for multicast is challenging, because providing reliability for each host needs special attention.

TCP-SMO [3] is simple approach to extend TCP to perform multipoint data delivery. This approach was called Singlesource- Multicast-Optimization (SMO), because TCP-SMO can be viewed as a simple optimization over multiple unicasts, which is needed and sufficient for most multicast applications.

TCP-XM [4] focuses on combining both unicast and multicast simultaneously. Multicast transmission will always be attempted, but equally, fallback to unicast transmission will always be available. It uses native TCP for control connections and TCP-XM (over UDP) for data transfers.

2.3.2 Problems and Solutions of DCCP multicast

DCCP endpoints progress through different states during the course of a connection, corresponding roughly to the three phases of initiation, data transfer, and termination.

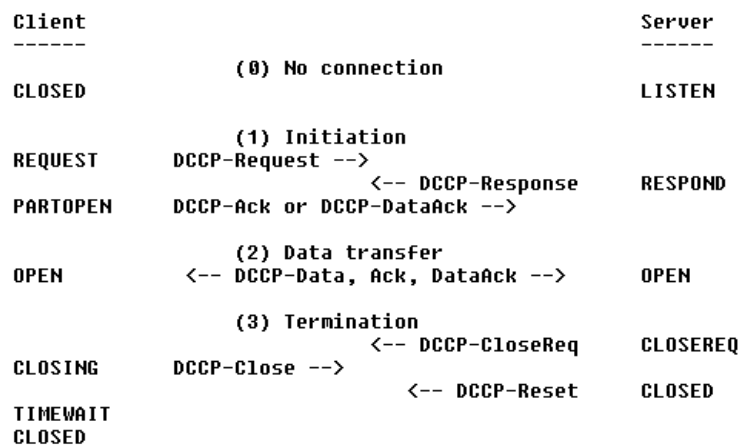


Figure 3 – Initiation, Data transfer, Termination with states

The DCCP protocol is connection-oriented; therefore the communication between the server and the client starts with connection setup (3-way handshake) as presented in Figure 3.

2.3.2.1 Protocol states

Usually the server is in LISTEN state while waiting for connection initiated by the client, which means that the client will send the first DCCP-Request message. In case of multicast scenario this is not acceptable, because each client will send DCCP-Request message. Using multicast, the clients must be in LISTEN state and waiting for the server. The server will send one DCCP-Request message that will be received by all the clients.

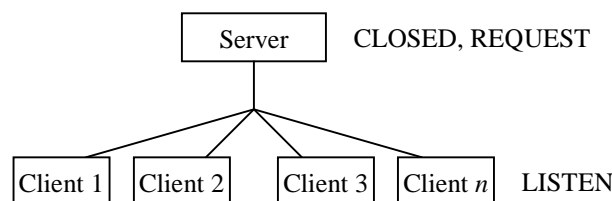


Figure 4 – Client, server states

2.3.2.2 Three-way handshake, sequence numbers

The DCCP-Request packet specifies the client and server ports, the service being requested, and any features being negotiated, including the CCID. As a response a DCCP-Response packet is sent. This response indicates any features and options that are accepted. In the third phase a DCCP-Ack packet is sent that acknowledges the DCCP-Response packet. This acknowledges the initial sequence number.

If multicast scenario is assumed, the server sends the DCCP-Request packet with randomly generated sequence number field and all the clients will answer with DCCP-Response packets. The problem is that each DCCP-Response packet will have different sequence numbers, but the server can acknowledge (DCCP-Ack) only one of them.

The solution should be to have the same sequence number for each client's DCCP-Response packet and keep only one client's DCCP-Response packet. Unfortunately the DCCP have some security considerations and it does not allow manipulating the sequence numbers. By modifying the DCCP kernel implementation this problem can be solved. A

simple solution should be to initialize the both the server's and client's sequence numbers, e.g. setting the sequence number to 0 before the connection. To make this, the `linux/net/dccp/ipv6.c` and `linux/net/dccp/ipv4.c` files must be modified:

```
dreq->dreq_iss = dccp_v6_init_sequence(skb); // dreq->dreq_iss = 0;
dreq->dreq_iss = dccp_v4_init_sequence(skb); // dreq->dreq_iss = 0;
```

2.3.2.3 Synchronization

Any DCCP endpoint might receive packets that are not actually part of the current connection. For instance, the network might deliver an old packet, an attacker might attempt to hijack a connection, or the other endpoint might crash, causing a half-open connection. DCCP, like TCP, uses sequence number checks to detect these cases. Packets whose Sequence and/or Acknowledgement Numbers are out of range are called sequence-invalid and are not processed normally. DCCP requires a synchronization mechanism to recover from large bursts of loss. DCCP uses DCCP-Sync and DCCP-SyncAck to synchronize the sequence numbers after receiving a packet that is not actually part of the current connection. On receiving a sequence-valid DCCP-Sync packet, the peer endpoint must update its GSR (Greatest Sequence Number Received) variable and reply with a DCCP-SyncAck packet. The DCCP-SyncAck packet's Acknowledgement Number will equal the DCCP-Sync's Sequence Number, which is not necessarily GSR.

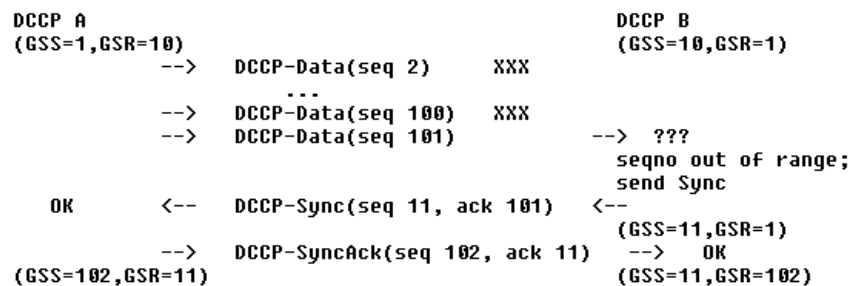


Figure 5 – DCCP synchronization

For our purposes this synchronization mechanism can be utilized to set the clients' sequence numbers to a given value. If the server sends a DCCP-Sync(seq X , ack Y) packet, all the clients will respond with a DCCP-SyncAck(seq $Y+1$, ack X). After the synchronization, the server can send DCCP-Data(seq $X+1$) data packet, while all the clients will answer with a DCCP-Ack(seq $Y+2$, ack $X+1$). With this solution all the clients will have the same sequence number.

The new arising problem is: how to force the server to send a DCCP-Sync packet without modifying the DCCP implementation. According to RFC4340, any sequence-invalid packet must elicit a DCCP-Sync packet, but if the peer endpoint is in the REQUEST state, it must respond to the DCCP-Sync packet with a DCCP-Reset instead of a DCCP-SyncAck. Unfortunately the sequence number synchronization can be done only after the 3-way handshake process.

2.3.2.3.1 Sequence window

Each DCCP endpoint defines sequence validity windows that are subsets of the Sequence and Acknowledgement Number spaces. These windows correspond to packets the endpoint expects to receive in the next few round-trip times. The Sequence and Acknowledgement Number windows always contain GSR (Greatest Sequence Number Received) and GSS (Greatest Sequence Number Sent), respectively. The window widths are controlled by Sequence Window features for the two half-connections. If the Sequence Window is W packets wide the valid sequence numbers are considered as presented in Figure 6 below.

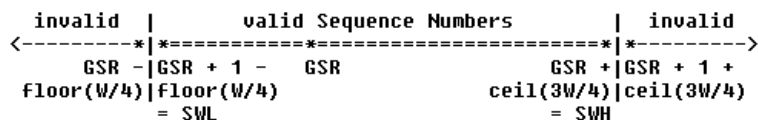


Figure 6 – Valid Sequence numbers

The Acknowledgement Number validity is considered similarly:

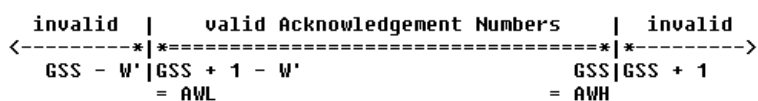


Figure 7 – Acknowledgement Number validity

By setting the default sequence window to its maximal value the synchronization probability can be reduced. New connections start with Sequence Window 100 for both endpoints.

Sequence Window is non-negotiable and takes 48-bit (6-byte) integer values, like DCCP sequence numbers. The maximum valid Sequence Window value is:

$$W_{MAX} = 2^{46} - 1 = 70368744177663$$

Changing the Sequence Window can be done with `sysctl` command in Linux:

`sysctl -w net.dccp.default.seq_window=WMAX`.

While the Sequence Window maximal value is equal to DCCP sequence numbers, all received packet will be valid, if $W_{MAX}=2^{46}-1$ is used. In this case there will be no need for synchronization and no packets will be drop due to invalid sequence number. This setting must be done for both client and server side.

This behavior of Sequence Window can be utilized in the multicast scheme, because in this case the clients use different sequence numbers according to the standard. All the packets will be considered as valid, if the Sequence Window is set to its maximum.

2.3.2.4 Multiple packets

DCCP is not a reliable protocol, however, it uses acknowledgement to get information about the lost packets. DCCP's Acknowledgement Number field is equal to the greatest sequence number received.

In a multicast scenario each client will send DCCP-Ack packets with the last packet's sequence number. Due to different link characteristics the client will not receive the servers DCCP-Data packets at the same time. DCCP-Ack packets from a different source, with different acknowledged sequence numbers will cause the disconnection.

The solution is to use packet filtering and allows only one client to deliver its DCCP-Ack packets. Packet filtering can be easily done using *iptables* (it already supports DCCP).

The problem of multiple DCCP-Response packets is similar, as well the solution.

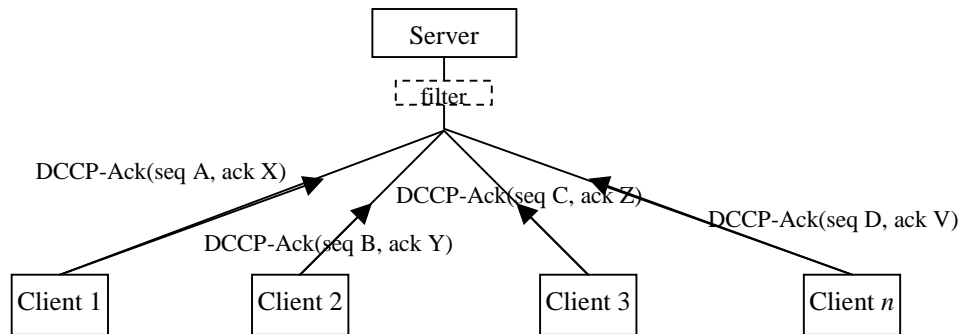


Figure 8 – Multiple DCCP-Acks

2.3.3 Proposed DCCP multicast model

In the previous section we have collected numerous difficulties caused by multicast delivery of packets and also proposed some solutions to the mentioned problems. In this section we present a guideline, that is how to set up DCCP for multicast.

In order to avoid DCCP source code patching, we can use the Sequence Window feature of the protocol instead of modifying the initial sequence number to e.g. 0. The first step is to set the Sequence Window to 70368744177663 at the server and the client. This is the key to make clients ready to receive DCCP packets sent for the assigned client.

`sysctl -w net.dccp.default.seq_window = 70368744177663`

With this setup all packets are considered valid, therefore no synchronization process is needed.

In the proposed multicast scheme the clients are in LISTEN state waiting for the appearance of the server. The server will send the first DCCP-Request message to IP_{MC} multicast address. The sent request will be received by all clients joined to the multicast group. The server can communicate with only one client, call it assigned client. In Figure 9, Client 1 is the assigned client. The packet sent by the other clients (Client 2 and Client 3) must be filtered. There is no rule how to choose from the clients and dedicate one as the assigned client, but selecting the sender of the first received DCCP-Response is a simple solution.

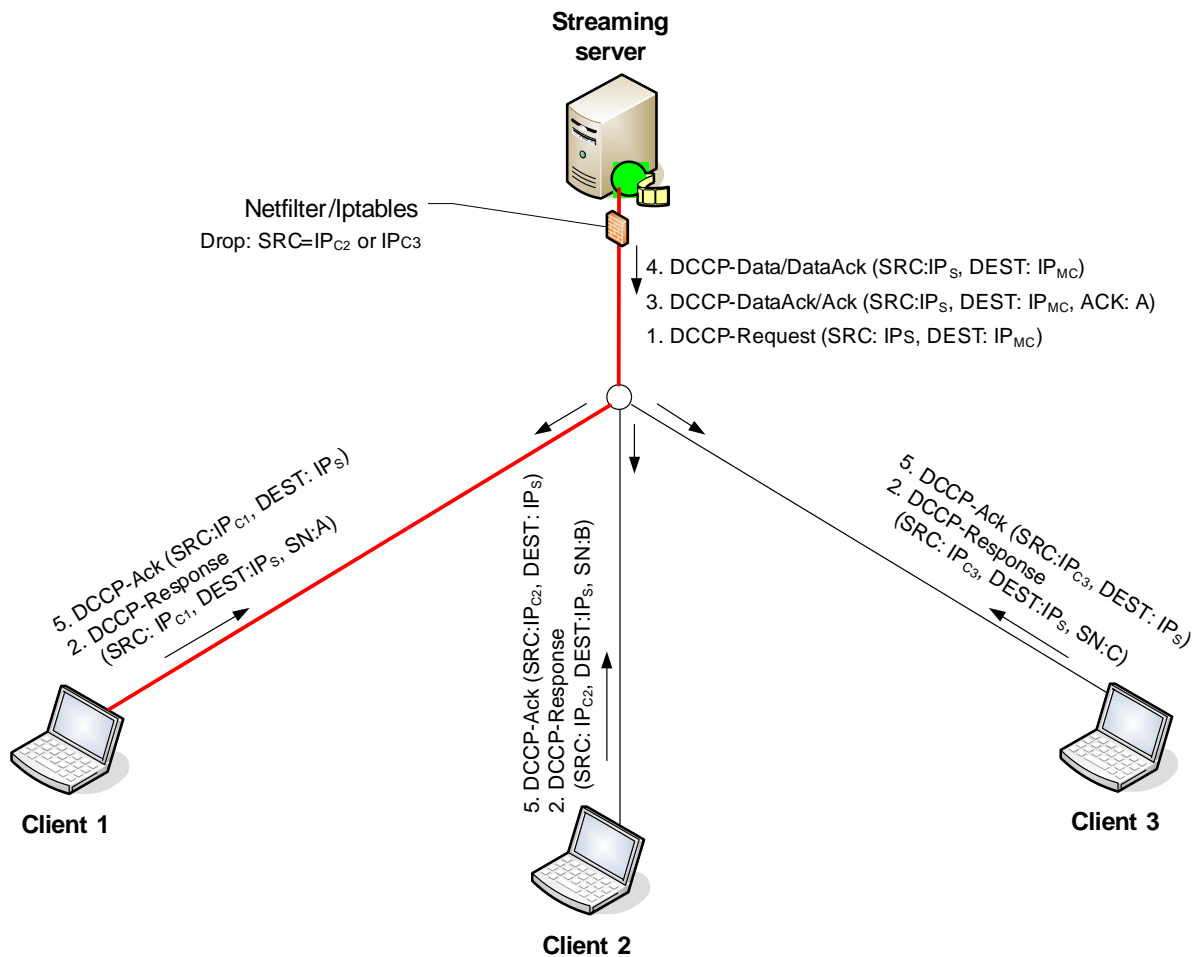


Figure 9 – DCCP multicast scheme

To drop the unwanted packets, the Linux kernels provide packet filtering. Using *iptables* command the packets of other clients can be dropped:

```
iptables -t filter -A INPUT -s IPC2 -j DROP
iptables -t filter -A INPUT -s IPC3 -j DROP
```

The server will communicate with the assigned client (Client 1), but the DCCP-Data and DCCP/DataAck sent by the server will be received by all the clients. In the DCCP-DataAck packets the server acknowledges the DCCP-Ack messages received from the assigned client. If the Sequence Window is set to its maximal value it should not cause problems.

While the DCCP packets of the clients, except the assigned one, are dropped, the disconnection of the assigned client will close the DCCP socket at the server. The other clients will receive only a copy of the packets sent by the server to the assigned client.

2.3.4 Benefits and Enhancement of DCCP multicast

Using multicast delivery of packets will significantly reduce the traffic load on the common links from the server to the clients. Using the proposed DCCP multicast scheme the server will adjust the sending rate to the link conditions between the server and the assigned client, because the server receives only the assigned client's DCCP-Ack messages. If the DCCP-Ack packets of other clients are not dropped but processed, link information about the other branches of the multicast tree can be obtained. Based on the separately handled DCCP-Ack messages the server could adapt its sending rate accordingly. Using TFRC [5] congestion control for DCCP, the DCCP-Ack/DataAck messages contains a Receive Rate [Byte/s] field. Based on this parameter, the server could decide the sending rate. However, it raises several open issues.

First of all, the policy of service, or in other words, how the multiple feedback parameters are processed should be clarified. If all clients must be served, the lowest (worst) rate must be used for sure. If the best user is selected, then we can provide the best quality to this user, but all other users will receive lower quality streams, due to frequent packet losses. Anything between these two extreme mechanisms (the average or somehow calculated) yields a compromise between quality and the number of successfully served clients.

Secondly, the Receive Rate is a stochastic variable: it changes continuously. The question is how to handle the stochastic nature: moving average or probabilistic distribution modeling could be options here.

2.4 DCCP feedback management

In case of multicast, each client may have different needs and capabilities, but only one can be taken into consideration. The congestion control algorithm can accept only one source of feedback packets at the same time. The multicast group is not static; therefore some clients may leave the group, while others may join to it. The arising questions are:

- Which client's DCCP-Acks should be received by the server?
- How to change the selected client during the communication?
- What happens if the selected client leaves the multicast group?
- What happens if a new client joins the multicast group?

The answer is to use an intelligent packet filter, which adaptively selects the DCCP-Ack feedback packets and forwards the selected ones to the server. To do this, all the DCCP feedback packets must travel through an *aggregation server*, where the packet filtering is done. The aggregation server is not necessary an independent device, but it can be deployed in the server too. The task of the aggregation server is to handle the incoming DCCP feedback packets, select the appropriate client, which DCCP-Ack packets are forwarded to the server. While DCCP is a connection oriented protocol, the aggregation server must also manage the IP header's source and destination fields. The IP addresses must apparently be remained unchanged during the communication. The *iptables* in Linux is a preferable tool to manipulate the IP source and destination addresses.

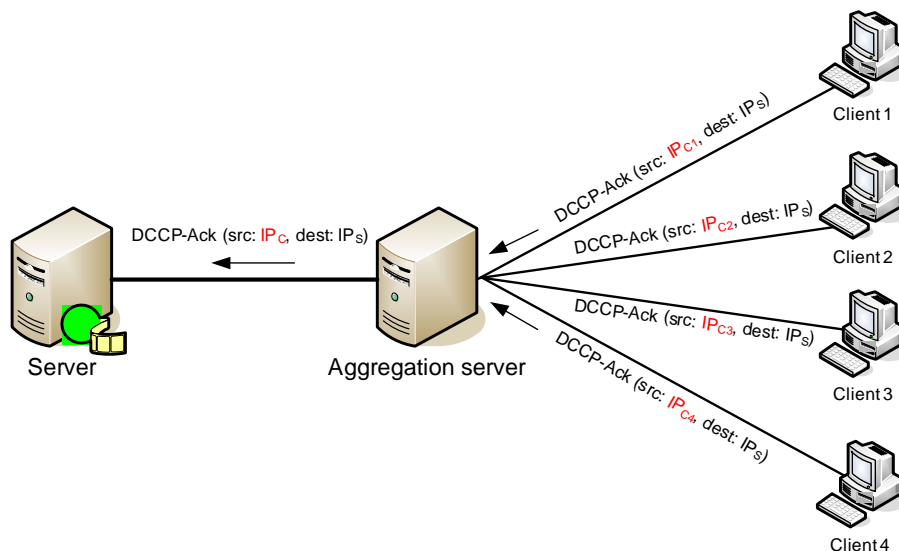


Figure 10 – DCCP-Ack aggregation

The aggregation server must set the IPC address to be equal to any of the clients' IP addresses or it can be even a new address if address translation is used. It is important that the IP addresses appearing at the server in the IP header must not change, therefore we propose to use a new source address (IP_C) in the DCCP-Ack packets' IP header delivered from the Aggregation server to the Server. In this case there will be no difficulties, if the selected client leaves the multicast group, because the Aggregation server can seamlessly select another client. The sequence of sent and received packets is presented in Figure 11, where Client 1 is the selected client and the DCCP-Ack packets of all other clients are dropped by the aggregation server.

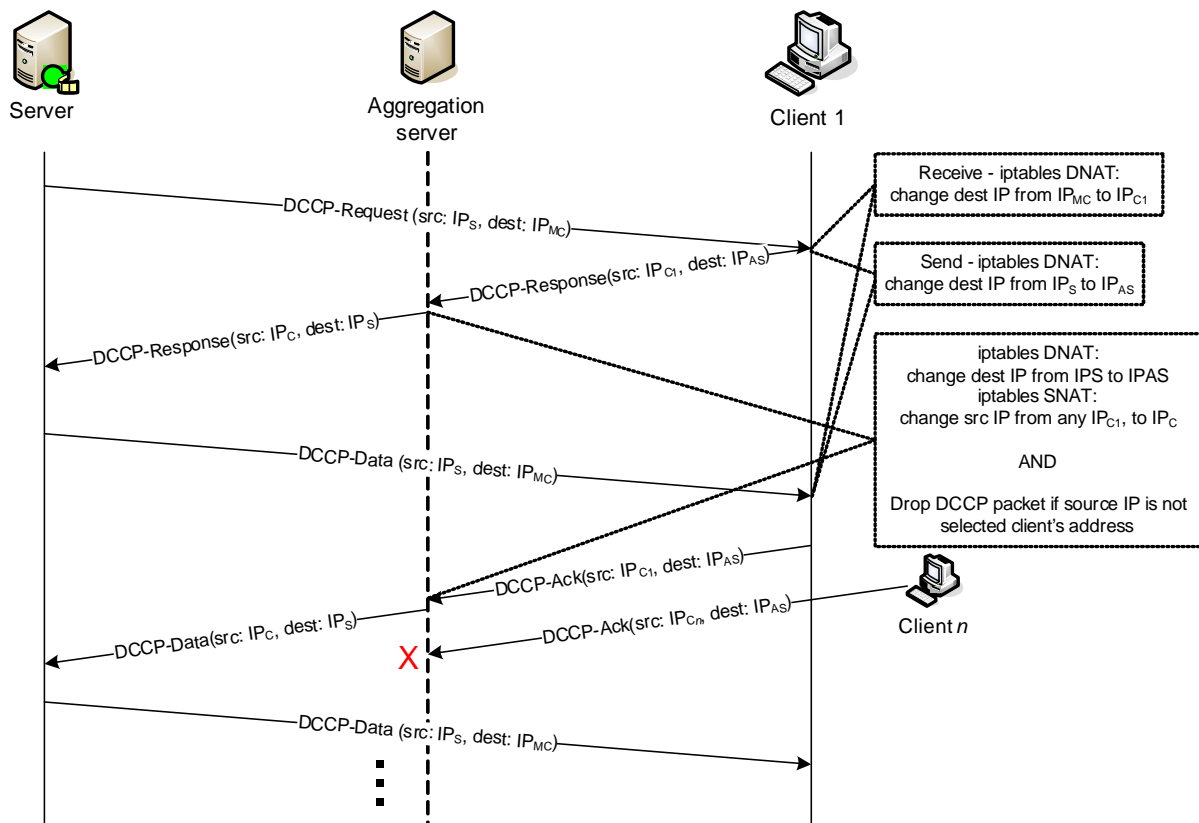


Figure 11 – DCCP packet sequence chart

2.4.1 Selecting client at connection setup

In the phase of connection setup the server sends a DCCP-Request packet for the clients in the multicast group. In this phase there is no information about the number and capabilities of the clients, however the response time provides some support for selecting the client first time. We can assume that the first arriving DCCP-Response determines the link with the lowest Round Trip Time (RTT). Generally low RTT belongs to a good quality link, supposing higher bandwidth. The selected client will determine the sending rate of the server. Unfortunately the RTT can be independent of the available bitrate of the link, but in the beginning of the connection no other information is available. To select the client, which DCCP feedback packets will be forwarded to the streaming server, different strategies can be followed at the connection setup. The methods are described in the following subsections.

2.4.1.1 Worst link

If the goal is to serve all the clients joined to the multicast group and minimize the overall packet loss rate due to overloading, the client with the slowest link should be selected. We assume that the link quality is in correlation with the RTT, so the IP header of the last received DCCP-Response will determine the client to be selected. In this case the chosen client will receive the data with the highest bitrate possible, while all the other clients will have free rate capacity. The advantage of this solution is that no packet will be lost due to overloading of a client or its link.

2.4.1.2 Serve given number of clients

If some clients are served with higher bitrate without overloading, some clients will not be able to receive the transferred data on that rate. If there are N clients in the multicast group and our aim is to serve n ($n \leq N$) number of clients without overloading them, we have to select the client, which DCCP-Response packet was received n^{th} . The other $N-n$ clients will have higher loss rate, because the sent data can not be received on the applied bitrate. The overall loss rate (π) of the multicast group due the increased bitrate will be

$$\pi = \frac{1 - \sum_{i=n+1}^N \frac{\lambda_i}{\lambda_n}}{N}$$

where λ_i is the receive rate of i^{th} client (the order is based on the receive order of DCCP Response packets) and λ is the receive rate of the selected (n^{th}) client.

2.4.1.3 Maximum bitrate

The aggregation server may forward the first arriving DCCP-Response packet and discard all others. In this case the sending rate will be probably the highest if we assume that the RTT and the link capacity are in inverse proportion. The packet loss due the high transfer rate will be the highest, because only one (first) client will be able to receive the sent packets without loss. The overall loss rate due to overloads will be

$$\pi = \frac{1 - \sum_{i=2}^N \frac{\lambda_i}{\lambda_1}}{N}$$

2.4.2 Change client during the communication

The client, which determinates the sending rate of the server with its DCCP-Ack packets, can be easily changed during the data transfer. Between the server and the aggregation server only IP_S , IP_C , IP_{MC} addresses are used that are all independent from the clients' IP addresses (IP_{Cn}). The aggregation server manipulates the IP addresses by setting IP_C to one of the clients' address and drops the DCCP-Ack and DCCP-Response packets of other clients. To change the client during the communication IP_C must be set to the selected client's IP address, moreover the packet filter at the aggregation server must be modified to drop all DCCP feedback packets except ones belonging to the selected client. To select the new client different strategies can be used:

2.4.2.1 Connection setup like

The first method is to utilize the DCCP-Response receive order at connection setup and select the client according to the previously presented three different methods (1. Worst link, 2. Serve given number of clients, 3. Maximum bitrate). The disadvantage of this solution is that the link and client capabilities may change in time, so the acceptable receive rate of the client is also changing.

2.4.2.2 RTT measurement based

The second solution is to continuously measure the link quality by RTT measurement and maintain a link order list. This solution solves the problem of the first one, but needs extra ICMP messages for the RTT measurements.

2.4.2.3 DCCP-Ack monitoring (TFRC)

The third technique is based on DCCP-Ack monitoring if TFRC congestion control algorithm is used. Using TFRC the DCCP-Ack packets contains a receive rate field in the DCCP header. The client that can receive the data packets without losses due to overloading will send DCCP-Ack with receive rate field equal to sending rate of the server. Clients with lower capabilities will have lower receive rates than the server sending rate.

Table 1 – Example of receive rates of the clients (sending rate is 500kbps)

| Host | kbps |
|----------|------|
| Client 1 | 500 |
| Client 2 | 500 |
| Client 3 | 500 |
| Client 4 | 267 |
| Client 5 | 159 |

The aggregation server must maintain a database of receive rates based on the DCCP-Acks of all clients, however only one client's feedbacks are allowed to be forwarded to the server. In the example only three clients (Client 1, Client 2 and Client 3) are served without overload loss. Based on the continuously refreshed receive rate database we propose an algorithm to serve n ($n \leq N$) number of clients without loss, where n is user defined.

2.4.2.3.1 Serve n algorithm

Based on the receive rate database there are three possibilities regarding to the actual number of served clients (n_s) before the new client selection:

1. $n_s < n$

The number of served clients has to be increased by selecting the n^{th} client from the database ordered by receive rate. Example (Table 1): If we want to serve $n=4$ clients, but currently $n_s=3$ clients are served, the 4th client (Client 4) must be chosen. By selecting Client 4, the server's sending rate will be 267kbps, while the receive rates of Client 1 to Client 4 will be 267kbps. Client 5 will remain unchanged.

2. $n_s = n$

Nothing to change.

3. $n_s > n$

Unfortunately only the current receive rate of a client is known and not the maximal receive rate it can reach. If we want to decrease the number of served clients by increasing the sending rate, we have to choose another client with higher maximal receive rate. This can be done only by testing the clients whose receive rate is equal to the server's sending rate. Example (Table 2): If we want to serve $n=1$ (highest receive rate), but currently $n_s=3$ clients are served, we must select all Client 1, Client 2 and Client 3 to find the client with the highest receive rate. For example, when the appropriate client (e.g. Client 3) is found and the server increases the sending rate based on the DCCP-Acks, the receive rate database can be filled as follows:

Table 2 – Another example of receive rates of the clients (sending rate is 500kbps)

| Host | kbps |
|----------|------|
| Client 1 | 500 |
| Client 2 | 870 |
| Client 3 | 1000 |
| Client 4 | 267 |
| Client 5 | 159 |

2.4.3 Selected client leaves the multicast group

During the multicast delivery of DCCP-Data packets any of the clients may leave the multicast group. If the selected client wants to do this, new client must be selected, which DCCP-Ack packets are allowed to reach the server, while others are filtered. The method is the same to the case when the client was changed during the communication, with the exception that the leaving client is not in the receive rate database.

To recognize that a client is leaving the multicast group, the IGMP messages or the DCCP-Close messages must be monitored.

2.4.4 New client joins the multicast group

DCCP is an anycast, connection oriented protocol, therefore connection setup is needed before the user data delivery, otherwise no data delivery is possible. During the data transfer the server can not establish new connection with the same service code. The service code is set by the server and it is sent within the DCCP-Request packet. To solve the problem, a DCCP-Request packet must be generated, which is identical to the original one sent to the multicast address, and send it directly to the new client. Excluding the generated DCCP-Request packet, the new client must not receive any DCCP packet before sending DCCP-Response packet, which will be filtered by the aggregation server.

3 QoS over IP networks

3.1 Absolute QoS in IP network

In the OPTIMIX architecture the data exchange between media servers and terminals goes through an IP network: this network must be able to transport any type of traffic, both real time and not real time, and also to support Quality of Service (QoS), i.e. throughput, error rate, end-to-end delay and jitter with the final objective to improve the user perception of the service. This aim is achieved implementing a QoS architecture in the routers of the IP network using one of the standard solutions proposed by IETF: Internet Integrated Services (IIS) or Internet Differentiated Services (IDS) [6].

In [7], we have introduced for the first time the concept of providing absolute QoS in an IP network relying on a relative QoS model, based on a Proportional delay Differentiation Model (PDDM). The technique is applied only to the real time traffic, because the QoS requirements of this type of traffic are more stringent than those for the data traffic. In high load condition, this traffic can be promoted to a better service class in order to guarantee the required quality and therefore, satisfy the related subscribed Service Level Agreement (SLA). Furthermore, when the high load condition ends, the real time traffic can be downgraded back to its original service class, because that class is again able to meet the SLA.

The analysis was made considering the AWTP scheduling algorithm, studying different approaches for promotion and downgrading the traffic: a fixed threshold, a moving average and a low pass filter. In this section instead we consider the DWFQ algorithm, with the same approaches. The objective is to analyze and compare the promoting and downgrading issues investigating two critical aspects: stability of the traffic performance and the adaptation speed to the network load changes.

Later, we compare the DWFQ and AWTP algorithms in the best own settings for absolute QoS provisioning, in order to choose the proper technique to be used in OPTIMIX architecture, more specifically considering different network configurations and input traffic conditions. Finally we evaluate the impact of the best solution on the joint source-channel (de-)coding scheme proposed by the project.

3.1.1 Absolute QoS in a IP network: promoting and downgrading UDP traffic

As stated in [6] and [7], the end-to-end QoS support in the IP network of the OPTIMIX system architecture is offered using the IDS architecture with a relative differentiated service model, the Proportional Differentiation Model (PDM). This model allows having different service classes, with a proportional defined service level having a specific performance differentiation between classes on the basis of a pre-defined criterion (for example pricing). However the relative model only assures that, if a higher class is used, the received service is better, without giving absolute guarantees.

In [7] we have proposed a technique that allows offering an absolute QoS to real time flows even if the IP network can support only a relative quality model. This aim is achieved by promoting the selected traffic flow to a higher QoS service class, supposing that the quality parameters required by the flow can be guaranteed in that higher class. The promotion can be done only if the target quality parameters values, imposed by the subscribed SLA, aren't met by the current QoS class, due to traffic congestion. When the congestion finishes and the original class is able to meet the quality target again, the selected flow can go back to that class. We have selected only real time flows, because the real time traffic has more stringent quality requirements, in particular on end-to-end delay.

We selected the 99th percentile of the end-to-end delay of the real time traffic flow as the reference quality parameter for the promotion of the flow, since it represents one of the stronger requirements for the real time traffic (e.g. in case of voice 200 msec). Moreover the delay jitter is one of the major sources of packet losses [7].

The end-to-end delay is monitored either at the egress of the network or at the receiver terminal and its value is sent, using the triggering engine [9], to the ingress border router, which has previously registered to that trigger. At the beginning of this analysis we haven't taken into account the impact of the transferring delay (in Sections 3.1.1.1 and 3.1.1.2) for the network performance feedbacks, but in the subsequent detailed analysis (Section 3.1.1.3) we have not only considered the impact of the feedbacks delay on the network and user performance, but also the impact of a different quality parameter values and a greater number of nodes in the network

The measure of the 99th of the end-to-end delay must be done run-time, in order to decide for promotion as soon as the traffic conditions degrade. Our proposal is to store N_{udp} UDP packets delay values and then picking up the value that corresponds to the 99th percentile of the delay; that is, if N_{udp} is 100, we pick up the second greatest one.

When the feedback of the end-to-end delay arrives at the ingress border router, it is elaborated using one out of three different measurement process, described in the following and already presented in D3.3b [7]:

- **Moving average:** the ingress border router collects and stores K_{up} consecutive values of D_{99}^{udp} , the 99th percentile of the end-to-end delay of the UDP traffic, coming from the receiver; when a new value of the 99th percentile is received, the oldest value of the K_{up} consecutive values is discarded. Using these values the ingress border router calculates the moving average MA , that is,

$$MA = \frac{\sum_{i=0}^{i=K_{up}} D_{99}^{udp}}{K_{up}}$$

If MA is below a fixed threshold S (for example 10 msec), which represents the maximum delay that has been agreed in the SLA, the SLA is satisfied and nothing happens. Otherwise, if the moving average is greater than the fixed threshold S , it means that the SLA isn't satisfied and the UDP traffic needs to be promoted to the closest upper class.

- **Threshold:** the ingress border router checks if the 99th percentile of the end-to-end delay of the UDP traffic is greater than a fixed threshold S for K_{up} consecutive times. If this happens, the UDP traffic needs to be promoted.
- **Low pass filter:** the ingress border router uses the last calculated output for the averaged value of the 99th percentile of the end to end delay of the UDP traffic and the current one for calculating the new output of the low pass filter LPF , that is:

$$LPF = LPF_{old}^{udp} * P_{up} + D_{99}^{udp} * (1 - P_{up})$$

If the result is greater than a fixed threshold S , the SLA on the delay is considered not satisfied by the current class and the UDP traffic must be promoted.

If these calculations lead to the conclusion that the UDP traffic must be promoted in order to satisfy the SLA on the delay, the ingress border router assigns to the UDP packet flow the closest class with higher quality. If the UDP traffic experiments an end-to-end delay exceeding again the maximum allowed value, the traffic can be further promoted to the subsequent class, while if the traffic in the closest class with lower quality respects the maximum allowed end-to-end delay, the UDP flow can return to the previous class.

The decision of downgrading the UDP traffic is made with some calculation slightly different from those one used for promotion. In this case we monitored D_{99} , the 99th percentile of the end-to-end delay of the whole traffic for each class: if the UDP traffic has jumped from class 1 to class 2, the receiver registers the 99th percentile of the end-to-end delay of traffic of the class 1 (collecting N subsequent values of the end-to-end delay) and sends it to the ingress border router, which elaborates it according to one of these methods presented in [7]:

- **Moving average:** the ingress border router collects and stores K_{down} consecutive values of D_{99} , the 99th percentile of the end-to-end delay of the closest lower class; when a new value of the 99th percentile is calculated, the oldest value of the K_{down} consecutive values is discarded. Using these values, the ingress border router calculates the moving average MA , that is,

$$MA = \frac{\sum_{i=0}^{i=K_{down}} D_{99}}{K_{down}}$$

If MA is below a fixed threshold S (for example 10 msec), which represents the maximum delay that has been agreed in the SLA, the closest lower class has resolved the traffic load and the UDP traffic can return to that class. Otherwise, if the moving average is greater than the fixed threshold S , the UDP traffic remains in the better class (class 2 in the example).

- **Threshold:** the ingress border router checks if the 99th percentile of the end-to-end delay of the closest lower class is lower than a fixed threshold S for K_{down} consecutive times. If this happens, the UDP traffic needs to be downgraded.
- **Low pass filter:** the ingress border router filters, with a low pass filter LPF , the 99th percentile of the end-to-end delay of the UDP traffic:

$$LPF = LPF_{old} * P_{down} + D_{99} * (1 - P_{down})$$

If the result is lower than a fixed threshold S , the SLA on the delay is considered satisfied again by the closest lower class and the UDP traffic can be downgraded.

The monitoring of the end-to-end delay for each class is done independently from the promotion of the UDP traffic. This choice, even if quite stringent, can be assumed valid because we have considered negligible the movement of a small percentage of traffic between classes in a network scenario with high speed links. Even if this movement is not negligible, the choice remains valid because we have chosen to monitor for downgrading not the current class of the moved traffic, but the closest lower class.

In order to evaluate the best methods for promotion and downgrading, we have taken into account two factors: the stability of the system and the speed of reaction to the traffic load changes in the whole network. In fact, the system

must avoid continuously changing the assigned class, in order to guarantee the stability of the system, while on the other hand it must be fast enough to promote the traffic flow as soon as the end-to-end delay parameters don't meet the maximum allowed value anymore in the current class or downgrade the flow if the delay is lower than the reference quality parameter threshold in the closest class with lower service.

In Table 1, we summarize the configuration option for both the promotion and downgrading process.

Table 1 - Configuration parameters for promotion and downgrading UDP traffic

| Promotion | | Downgrading | |
|-------------------------|---|---------------------------|--|
| K_{up} | Number of consecutive values of the 99 th percentile of the end-to-end delay of the UDP traffic to be considered | K_{down} | Number of consecutive values of the 99 th percentile of the end-to-end delay of the traffic aggregates to be considered |
| N_{udp} | Number of consecutive packets on which the 99 th percentile of the end-to-end delay of the UDP traffic is calculated | N | Number of consecutive packets on which the 99 th percentile of the end-to-end delay of the traffic aggregates is calculated |
| S | Fixed delay threshold (as by SLA) | S | Fixed threshold (the same as for the promotion case) |
| P_{up} | Low pass filter pole | P_{down} | Low pass filter pole |
| <i>promotion method</i> | Moving average, threshold or low pass filter | <i>Downgrading method</i> | Moving average, threshold or low pass filter. |

3.1.1.1 AWTP with absolute QoS

In [7] we have analyzed a network scenario in which we have assumed to have an IDS architecture where routers are equipped with AWTP scheduling algorithm and support the absolute QoS techniques, which have been explained above. We have compared the different methods for promoting the UDP traffic flows and we have concluded that the moving average with $K_{up}=2$ method is able to guarantee a rapid reaction to the network traffic increase, also in case of traffic congestion, but fairly preserve the stability of the system. Instead, the downgrading process is best performed by threshold method with $K_{down}=20$, since the simulation results have shown that the threshold is the most conservative downgrading approach and hence the most stable: this method can quick react to significant network traffic changes, but it prevents the system from following the network traffic small oscillations.

3.1.1.2 DWFQ with absolute QoS

The Weighted Fair Queuing (WFQ) scheduling algorithm works assigning a priority value or weight to each queue, which, consequently, shares the transmission service proportionally according to this weight: the queue having highest priority receives preferential service than those with lower priority. WFQ cycles through queues, transmits bytes proportionally to the value of the weight for each queue, and fairly distributes the bandwidth allocated to not active queues again proportionally to the weights themselves.

the system, instead, uses the Dynamic WFQ (DWFQ), it adapts run-time the bandwidth allocation for each queue supporting a proportional differentiation model for QoS, like AWTP. In [6], we have proposed a new algorithm for weight updating in the dynamic WFQ scheduling discipline based on the Knightly's theory. Applying the absolute QoS approach, we have issued it in the following simulations.

3.1.1.2.1 Network scenario

The simulation analysis about DWFQ has been carried out in the same network scenario we have used for AWTP. It is composed of four sources of traffic and three routers, as depicted in Figure 12. Also in this case, the first router is able to (re-)mark and to classify the incoming traffic, while the remaining routers simply route the packets. Each router subdivides the incoming traffic in four queues, each one corresponding to a different Differentiated services class (the quality factors 1, 2, 3 and 4 are assigned to the four queues, respectively). Like in case of AWTP, each router discards about 50% of the received traffic, sending it to a node that acts like a trash, while the traffic that is generated by *Source 1* and *Source 2* and reach the receiver is measured and monitored.

The absolute QoS techniques can be applied only in the first router, which is able to promote or downgrade the UDP traffic, because it is considered as the ingress border router in a Diff Serv domain.

We set the link capacity between routers to 100 Mbit/s, while the links capacity between sources and routers is 500 Mbit/sec in order to avoid queuing at the source output buffer; furthermore we have assumed that the buffer size at the router output interfaces is set to infinity to avoid losses.

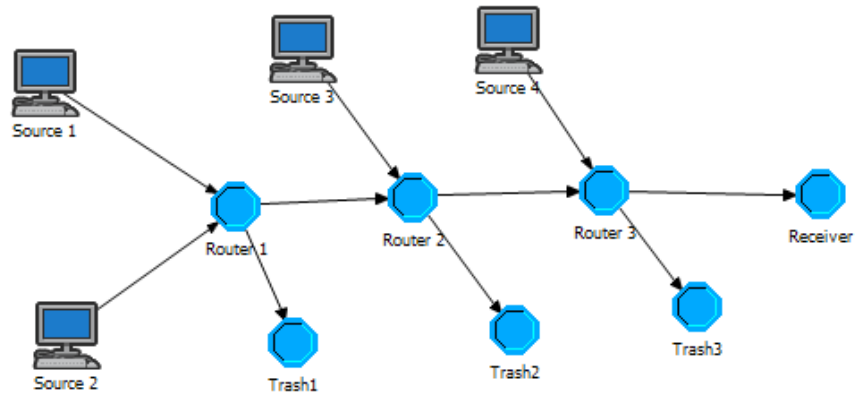


Figure 12 - Network scenario

In order to compare the behaviour of absolute QoS using AWTP and DWFQ, we have used with DWFQ the same aggregated traffic, which, recalling from [7], is composed of different real backbone traffic traces:

- 1 TCP source, with an average traffic rate of 16 Mbit/s,
- 17 video sources, each one with an average traffic rate of 280 Kbit/s,
- 3 audio sources, each one with a average traffic of 10 Kbit/s.

This traffic aggregation is replicated for each class of service and lasts for 60 seconds, reaching in total about 80 Mbit/s on average. During two time intervals (between 10 and 25 seconds and between 40 and 55 seconds) new traffic is added in order to create higher network load conditions, up to almost 100Mbit/s. This is exactly the same used in [7]

This added traffic is constituted by:

- A constant packet inter-generation time of 0.001 sec and a constant packet size of 4 Kbits between 10 and 25 seconds;
- A constant packet inter-generation time of 0.002 sec and a constant packet size of 7 Kbits between 40 and 55 seconds.

We have assumed that the traffic that is generated by the audio and video sources and the artificial one are all considered real time and uses UDP transport and it is about 20% of the whole aggregate; instead only part of the UDP traffic initially assigned to the worst class can be promoted or downgraded.

The first set of simulation parameters is reported in Table 3.

Table 3 – first set of configuration parameters

| | |
|---------------|---|
| $S=10$ ms | Fixed threshold (the value has been selected in order to properly check the correct operation of the proposal with a path composed of three routers) |
| $K_{up}=5$ | Number of consecutive values of the 99 th percentile of the end-to-end delay of the UDP traffic to be considered for promotion |
| $N_{udp}=10$ | The number of packets needed for calculating the 99 th percentile of the end-to-end delay for the UDP traffic. It is lower than N because the UDP traffic that can change class is only the 20% of the total traffic (therefore, a higher value would excessively slow down the system due to the delay introduced in the measurement process) |
| $P_{up}=0.9$ | Pole of the low pass filter for promotion |
| $K_{down}=20$ | Number of consecutive values of the 99 th percentile of the end-to-end delay of the traffic in an aggregate to be considered for downgrading the UDP traffic. The value is significantly bigger than K_{up} , since the traffic should return in the worse class only when the delay conditions have been better enough for a certain time |
| $N=100$ | The number of packets needed for calculating the 99 th percentile of the end-to-end delay for the traffic in each class |

$P_{down}=0.9$

Pole of the low pass filter for downgrading

3.1.1.2.2 Analysis of collected results and considerations

As we have done with AWTP we start comparing the three different methods for promoting the UDP traffic: our evaluation takes into account two factors, i.e. the stability and the reaction speed of the system to traffic changes, as we have highlighted in the above section.

We recall that all the figures represent the *dscp* (Diff Serv Code Point, i.e., the class) value of the UDP packets of the flow(s) that can change class: it starts from 1 and can reach value 4, which corresponds to the best class.

We start simulating the case of promoting the UDP traffic using the threshold method with $K_{up}=5$ and downgrading using the threshold method with $K_{down}=20$ (Figure 13). We can notice how the threshold method in promoting the traffic appears both slow and very conservative: the reaction speed is quite poor (the class change happens the first time 1.5 sec after the traffic change and after 6 seconds in case of the second traffic change), and the system doesn't follow the traffic oscillations. We have to take into account also a peculiar behaviour of the scheduler DWFQ that, in case of network congestion, tends to highly increase the delay of the first class without keeping the service proportionality between the different classes as dictated by the mutual ratio of the quality factors assigned to each queue. This means that there is a significantly gap between the delay of the first and second class, while the delays of the last three classes are more similar. As a consequence, after the class change due to the increase of the network load, the system stabilizes with the issued UDP flow on the second class, which is indeed able to satisfy the SLA without any other class change.

While considering the moving average method (Figure 14) for promotion, it appears slightly better: from the one side the promotion happens after 1.5 seconds at the first traffic increase, like the threshold case, while at the second traffic increase after only 4 seconds. On the other side, the methods show more reactivity to the traffic oscillations, better following also small traffic increments and decrements.

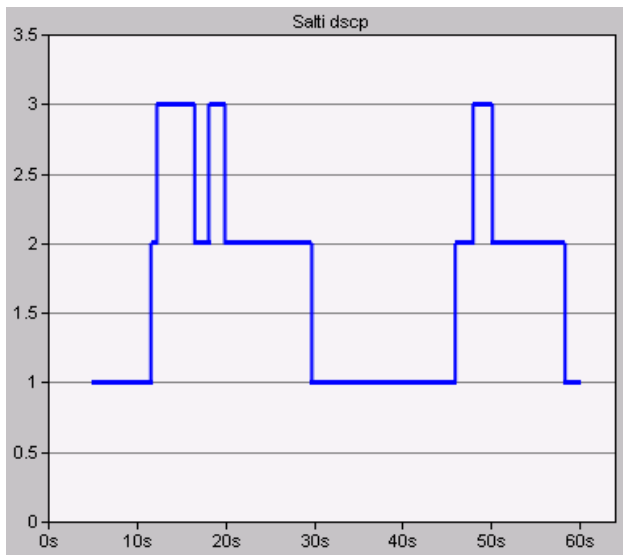


Figure 13 - Promotion by threshold with $K_{up}=5$, downgrading by threshold with $K_{down}=20$

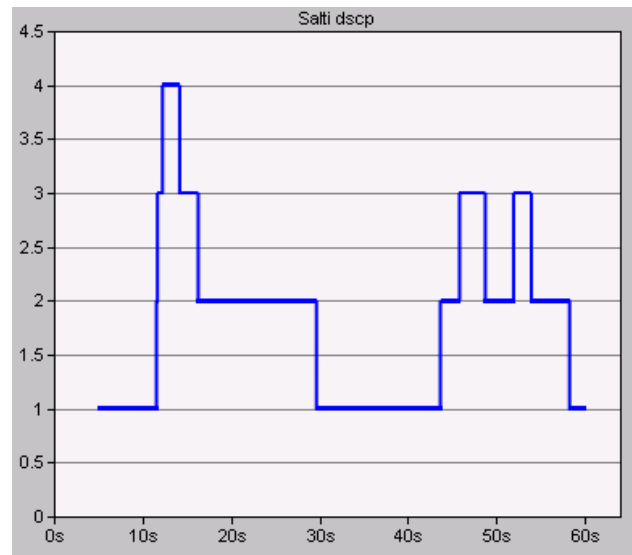


Figure 14 - Promotion by moving average with $K_{up}=5$, downgrading by threshold with $K_{down}=20$

If we consider the low pass filter method (Figure 15), also in this case the UDP traffic promotion at the first traffic increase happens after 1.5 seconds from the traffic change, while at the second traffic increase it happens after 3 seconds from the traffic change: for this reason this method seems better than the threshold and moving average methods. Moreover this last method better follows the traffic oscillations in particular during the second traffic increased after 40 seconds.

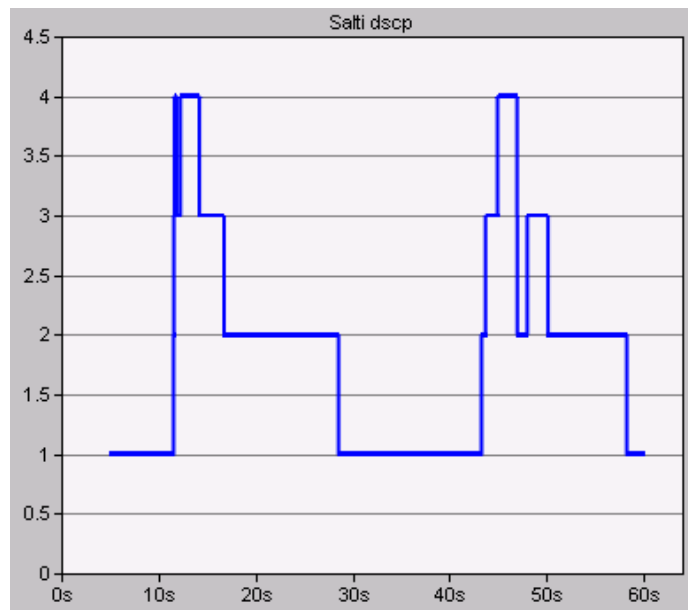


Figure 15 - Promotion by low pass filter with $P_{up}=0.9$, downgrading by threshold with $K_{down}=20$

Following the same approach we have adopted for AWTP, we have tried to optimize both threshold and moving average methods for promoting the traffic, reducing the number of samples of the 99th percentile of the end-to-end delay of the UDP traffic, to $K_{up}=2$.

With this new configuration the moving average method (Figure 16) doesn't reduce its reaction times, while the threshold (Figure 17) reduces only the reaction time at the second traffic change: now the promotion happens in 3-4 seconds like moving average and low pass filter. Regarding the reaction to traffic oscillations, significant changes are not detected in comparison with the case of $K_{up}=5$.

In general, we can conclude that when the network load increases we should adopt the low pass filter method, in order to have both a quick promotion of the real time traffic and a good reaction to traffic oscillations (stability).

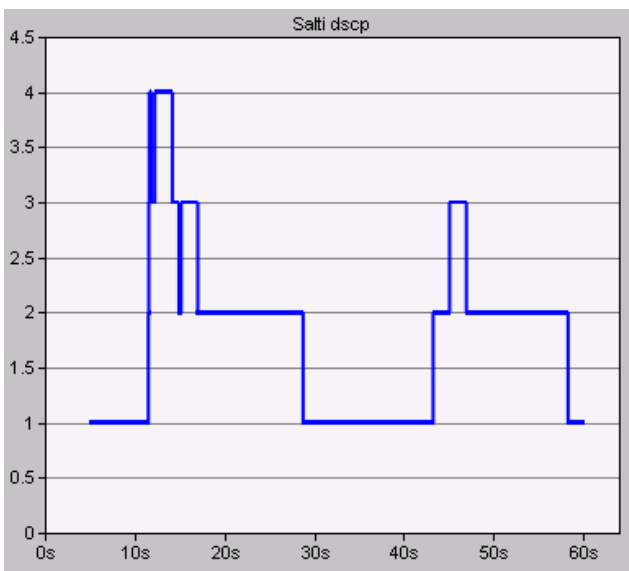


Figure 16 - Promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$

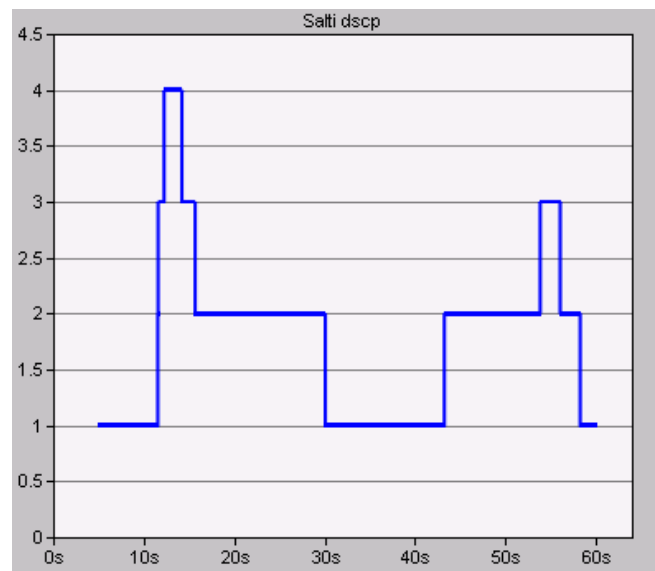


Figure 17 - Promotion by threshold with $K_{up}=2$, downgrading by threshold and $K_{down}=20$.

Let now fix the UDP traffic promotion method to the low pass filter and analyse instead the different methods for downgrading the UDP traffic. We have to evaluate both how fast the system reacts to a traffic reduction and the stability, which is how frequently the UDP traffic changes class when the network traffic oscillates. For this reason we have initially set a quite high value $K_{down}=20$ for the threshold and moving average methods.

Comparing again the three methods in downgrading and taking into account the above explained behaviour of the DWFQ algorithm, the threshold method appears to be more conservative (Figure 16), because the UDP traffic is downgraded when the traffic load reduces slightly and remains in the second class till the traffic increment finishes. This happens in both cases of the network load increase.

If we reduce K_{down} to 10, the system appears to be more reactive (Figure 18). Considering the first traffic change the system tries to follow the network load oscillations, and, as a consequence, there are frequent jumps between classes. However the reduction of K_{down} to 10 causes a not good reaction to the second traffic increase, because the system does not really follow the traffic oscillation.

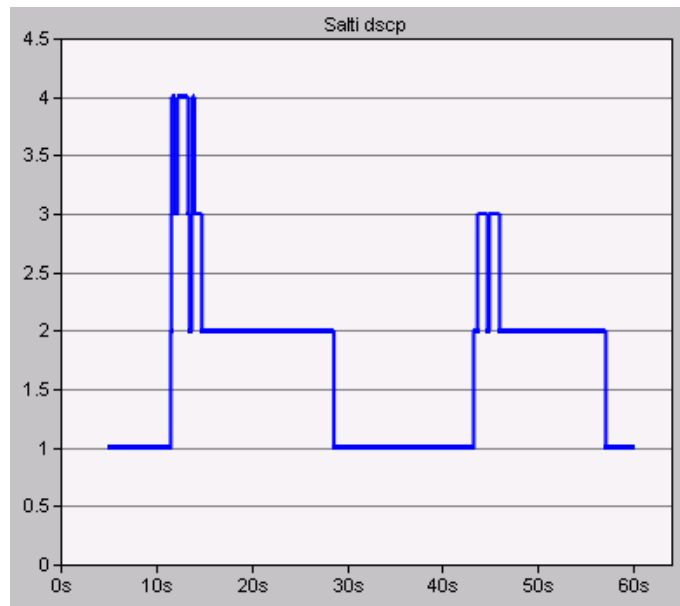


Figure 18 - Promotion by low pass filter, downgrading by threshold with $K_{down}=10$

On the other hand, if we consider the moving average method (Figure 21) we can notice that after the initial promotion of the UDP traffic, there is an immediate downgrading, followed by another promotion and so on. These oscillations happen every time there is a class promotion of the UDP traffic, like in case of AWTP. This behaviour can be explained by the fact that the calculation of the moving average is made with a certain number of samples of the 99th percentile of the end-to-end delay. At the beginning of the traffic change a lot of those samples are still under the fixed threshold and only a small number are above the threshold. As a consequence, while the UDP traffic needs to be promoted, the moving average of the 99th percentile of the end-to-end delay on the worse class can result for a period of time still below threshold and the UDP traffic is then downgraded. This situation causes observed oscillations. After this first period of transition, when the number of samples above threshold increases, also the moving average results above threshold and the UDP traffic isn't downgraded continuously any more. This explanation is confirmed by the fact that reducing the value of K_{down} the oscillations are not so frequent. Indeed, in this case (see Figure 20 and Figure 19), the number of samples that are below threshold for the moving average calculation reduces and consequently the number of oscillations.

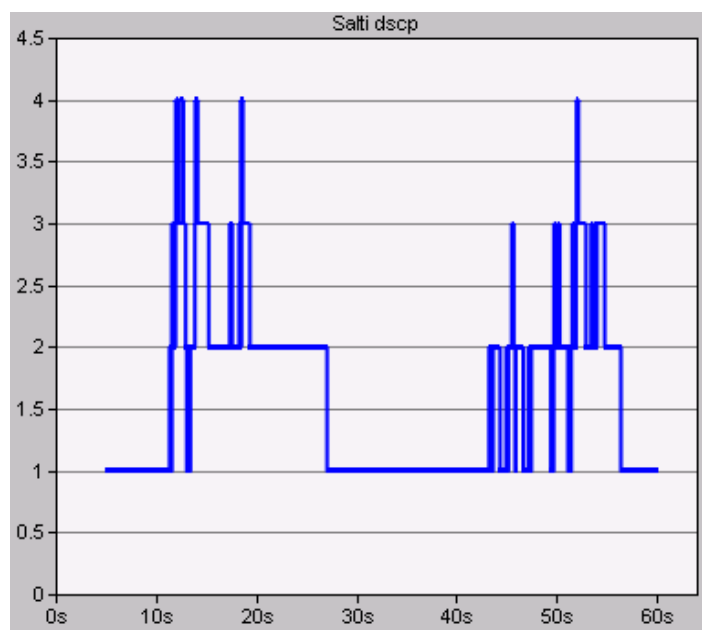


Figure 19 - Promotion by low pass filter, downgrading by moving average with $K_{down}=5$

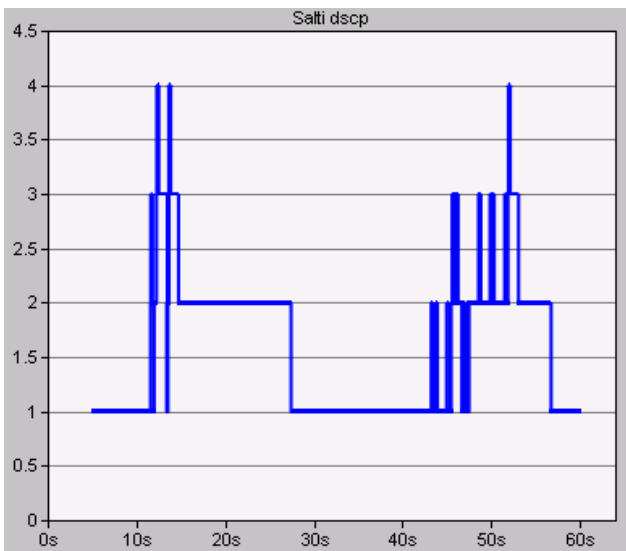


Figure 20 - Promotion by low pass filter, downgrading by moving average with $K_{down}=10$

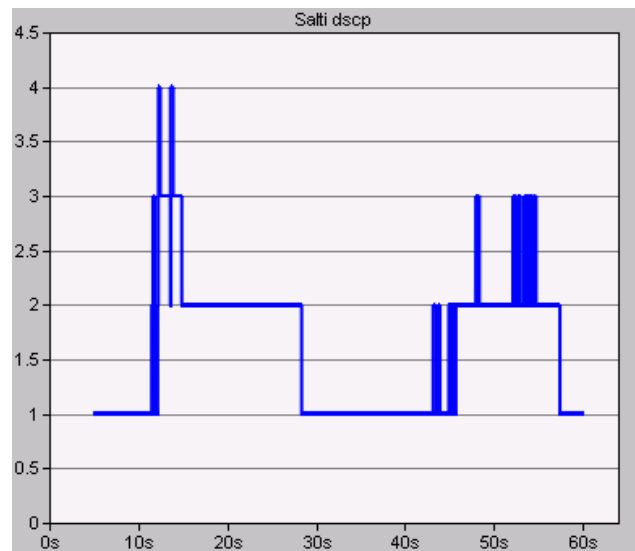


Figure 21 - Promotion with low pass filter, downgrading by moving average with $K_{down}=20$

Finally if we consider the low pass filter method (Figure 22), we can observe that there are some oscillations. These are more than in the case of moving average, in particular during the second increase of the traffic load. This is probably due to the fact that the system try to follow all the traffic dynamics very well, resulting in a complete instability and in continuous class changes. However even with a change in the pole, from 0.9 to 0.8 (Figure 23), the system behaviour does not improve significantly.

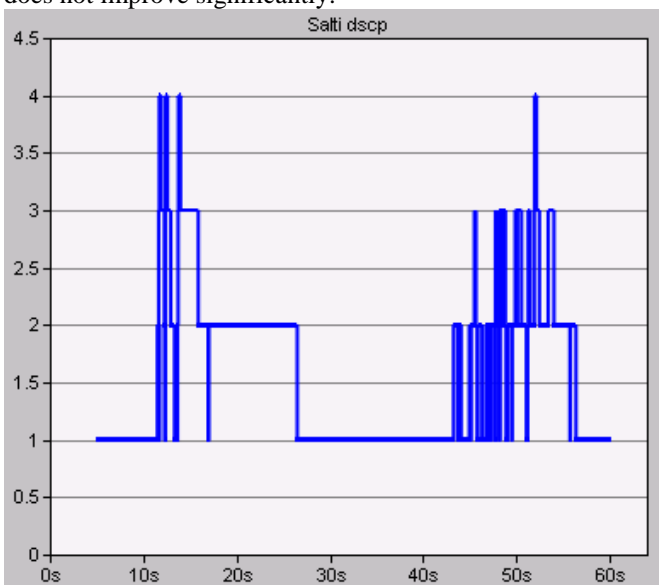


Figure 22 - Promotion by low pass filter, downgrading by low pass filter with $P_{down}=0.99$

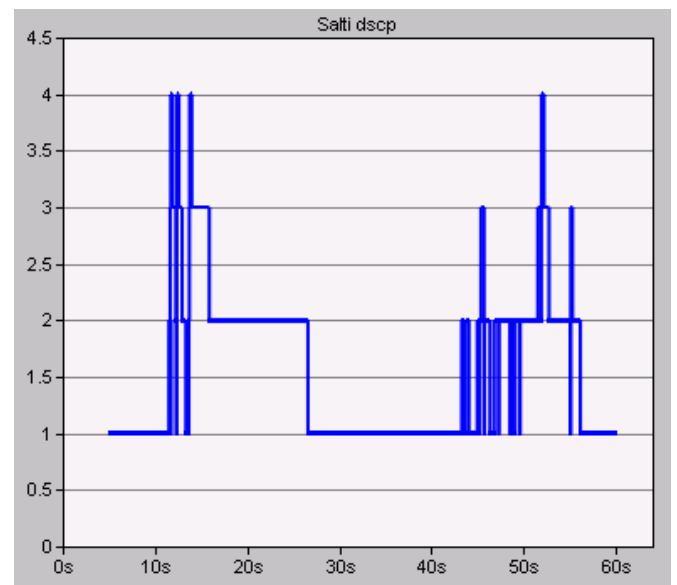


Figure 23 - Promotion by low pass filter, downgrading by low pass filter with $P_{down}=0.8$

Aiming to have a stable system but also fast reaction to traffic reductions and changes, we have to use threshold method with K_{down} set to a value between 10 and 20, according to the desired level of reaction to the traffic changes, for the downgrading of the real time traffic.

3.1.1.3 DWFQ vs AWTP for absolute QoS Support

Comparing the two considered scheduling algorithms, AWTP (in [7] and in Section 3.1.1.1) and DWFQ (in Section 3.1.1.2) aiming at supporting absolute QoS, we can notice that they show a similar behaviour. Considering the promotion methods, DWFQ appears more reactive than AWTP, showing very short delay between the traffic increase and promotion, in particular in the case of the second traffic change. However AWTP appears more reliable and consistent, because when the network load leads to congestion, the system reacts accordingly shifting the UDP traffic to the best class (class 4), while DWFQ instead saturates the worst queue and consequently the UDP traffic remains in the second queue because there the SLA is already satisfied. This situation is emphasised when threshold or moving average methods are used in promotion with $K_{up}=5$, while improves slightly if we reduce the value of K_{up} , in particular

at the second traffic change. As a consequence, we have chosen to use low pass filter with pole equal to 0.9 as promotion methods for DWFQ.

Of course, the saturation of the DWFQ scheduler, resulting in performance (i.e. delay) quite far from being proportional to the mutual ratio of the assigned quality factors, makes the scheduler a poor choice as to be deployed in a real network, where congestion phenomena can occur from time to time, even when a good planning of resource allocation and accurate SLA management have been applied.

Looking to downgrading methods, we find out that in both DWFQ and AWTP, the moving average and low pass filter show a lot of oscillations, because of their trying to follow any traffic change, which results in a high instability of the system. For this reason we select, also for DWFQ, the threshold method with K_{down} set between 10 and 20.

If we now compare the best methods for each scheduling algorithm, that is the moving average with $K_{up}=2$ for promoting and threshold with $K_{down}=20$ for downgrading in case of AWTP (Figure 24) and low pass filter with pole $P_{up}=0.9$ for promoting and threshold with $K_{down}=20$ for downgrading in case of DWFQ (Figure 25), we can see that AWTP follows in a smoother way the traffic dynamics without saturating the first queue, as DWFQ does. In fact it reacts quite fast when there is a load increase (the delays between the time of the traffic change and the promotion is similar to DWFQ) and in case of traffic congestion assigns the UDP traffic to the best class without many oscillations. This behaviour is confirmed if we look at the delay of the UDP traffic that can change class in both the above mentioned situations (Figure 26 and Figure 27 respectively).

As a consequence, we can conclude that the better solution to achieve absolute QoS in an IP network relying on a proportional relative model for QoS is to adopt the AWTP scheduling algorithm with moving average with $K_{up}=2$ for promoting and threshold with $K_{down}=20$ for downgrading. This choice takes also into account the conclusion we have obtained in [6], where we showed that AWTP behaves better than DWFQ since it maintains the proportional separation between classes in any traffic condition. Indeed, we must keep in mind that apart from the traffic that can change dynamically class of service to address its SLA, the remaining traffic within the network is granted with a service that is not absolute but proportional in accordance with the established price model or other agreements, which are strictly related to the mutual ratio of the quality factors assigned to each queue.

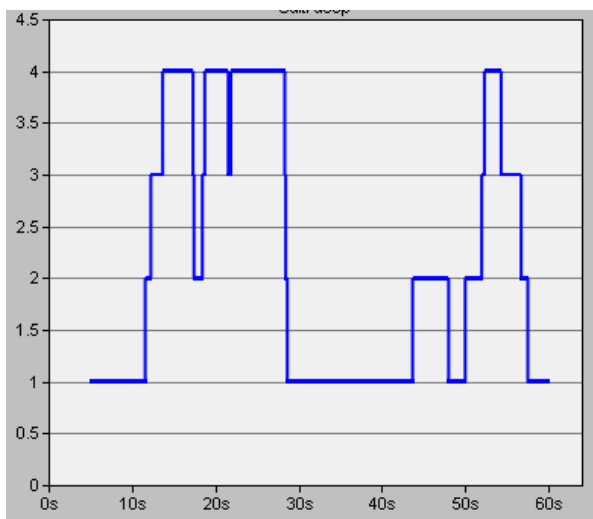


Figure 24 – AWTP, promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$

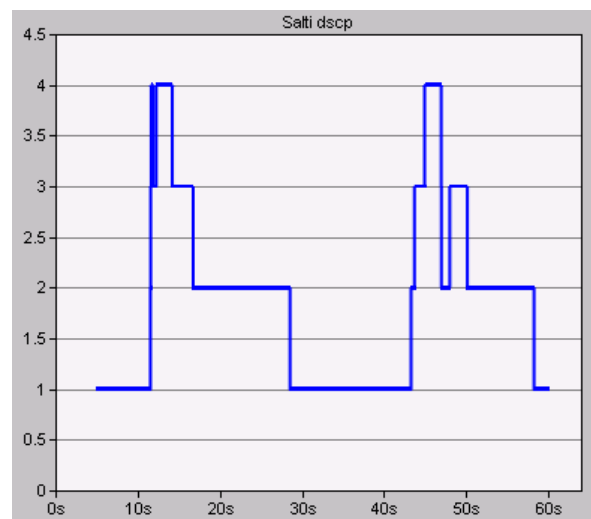


Figure 25 – DWFQ, promotion by low pass filter with pole $P_{up}=0.9$, downgrading by threshold and $K_{down}=20$

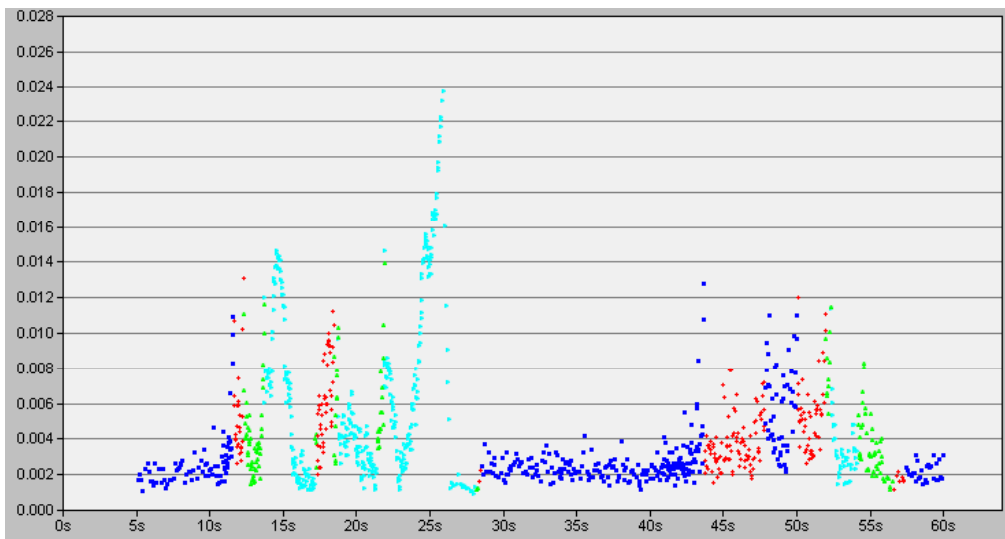


Figure 26 - AWTP, 99th percentile of the end-to-end delay of the UDP traffic: it starts in the first class (blue) than is promoted to the second class (red), to the third class (green) and finally to the best class (cyan). The threshold for promotion is fixed to 10 msec.

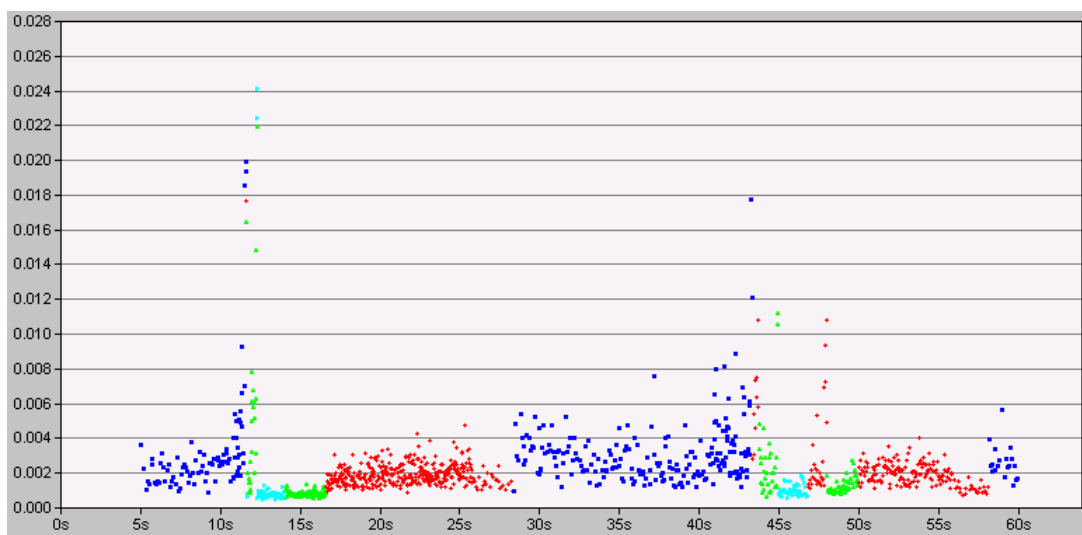


Figure 27 – DWFQ, 99th percentile of the end-to-end delay of the UDP traffic: it starts in the first class (blue) than is promoted to the second class (red), to the third class (green) and finally to the best class (cyan). The threshold for promotion is fixed to 10 msec.

3.1.2 Detailed analysis of AWTP to achieve absolute QoS

In the previous paragraph we have concluded that the best approach for supporting absolute QoS guarantees in an IP network relying on a proportional relative model for quality is to use the AWTP scheduling algorithm with moving average with $K_{up}=2$ for promoting and threshold with $K_{down}=20$ for downgrading. As a consequence we propose to deploy this option in a NGN, such as the telecommunication infrastructure referred by the OPTIMIX project.

In the OPTIMIX documentations, we have stated many times ([6] and [7]) that the IP network is QoS enabled, implementing the Internet Differentiated Services architecture. Figure 28 depicts the OPTIMIX IP network: the border routers of the DS domain allow the connection of the streaming server and the base stations to the IP core infrastructure. All the routers in the DS domain will use the AWTP algorithm for scheduling the traffic flows. In particular the DS Boundary Nodes (routers in green in the figure) have been enhanced in order to support the designed absolute QoS solution by the introduction of the described process of upgrading and downgrading the current service class of the issued real-time (multimedia) UDP traffic, by a measurement mechanism based on a moving average with $K_{up}=2$ for promoting and threshold with $K_{down}=20$ for downgrading. This means that these nodes receive the feedback regarding the 99th percentile of the end-to-end delay of the traffic for each class and then evaluate if the SLA is satisfied for the real time traffic. If the requirement on the delay is not addressed, the ingress node is able to remark the real time traffic in order to promote it in the better subsequent class. If during the transmission the SLA is satisfied again by the worse closer class, the router can remark the real time traffic in order to downgrade it to that class.

From an implementation point of view, it is necessary to identify the method that should be used to collect the end-to-end delays of the traffic flows. In the OPTIMIX environment we can imagine to have two different available approaches:

- Receiver-oriented: according to the OPTIMIX architecture [10] the triggering framework can be used in order to send the delay information. This means that the feedback is first sent to the Mobile Observer that will provide it to the ingress border router, if a node in the network has subscribed to the notifications of the feedback itself.
- Egress border router-oriented: we can suppose that the delay that is measured for one class between two border routers corresponds to the delay of every flows belonging to that class between those border routers. In this case we can imagine that the ISP measures the delays of the traffic flows between each couple of ingress and egress border routers and for every service class in order to verify if the SLAs with its customers are fulfilled or not. As a consequence the ISP can make the information on the delay available to the ingress routers in order to implement the proposed technique for absolute QoS guarantees relying on a proportional model of quality.

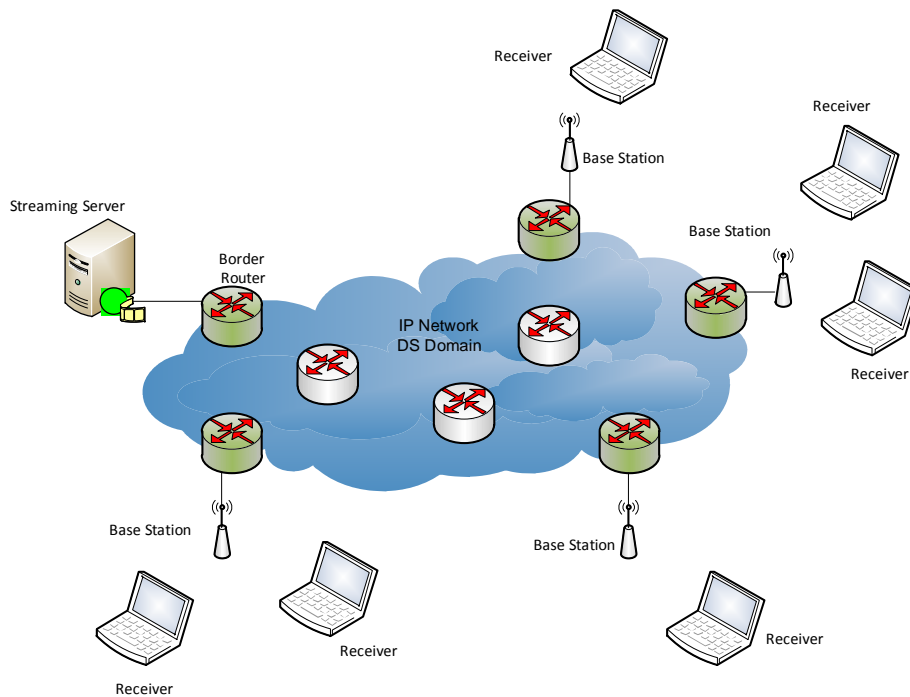


Figure 28 - OPTIMIX network architecture

In order to better understand what the performance of the designed solution for absolute QoS in the OPTIMIX environment are, we must deeply evaluate the two proposals. For doing this, we have considered three further scenarios:

- In a real situation the feedbacks need some time to arrive from the Mobile Observer or the monitoring node of the ISP to the ingress border router. We need to evaluate what is the impact of this delay on the absolute QoS solution.
- The traffic flow could go through a great number of nodes; in a first analysis we considered only three nodes; we consider now six nodes.
- In a first analysis we have set the quality factors of each issued schedulers to 1, 2, 3, 4 for the four service classes, respectively. We now investigate the behaviour with different values, for example increasing the distance between the classes: more precisely, the values 1,4,8,16 for the four service classes, respectively, are configured.

Moreover we have to properly evaluate the impact of our proposal on the OPTIMIX architecture. This can be done considering the resulting delay and delay-jitter of the real time traffic. The jitter in particular is very interesting because provides an idea of the magnitude of the reordering phenomena (possibly caused by the promoting operations). This is done looking at the jitter statistical distribution and considering the peak to peak distance, the 99th percentile and the variance of the delay. The jitter has been calculated as the difference between the values of the end-to-end delay of the UDP traffic that can change class and the mean of this delay.

However, it is worthwhile to point out that by a buffering mechanism at the egress border router and a proper DSCP marking, the re-ordering phenomena can be even avoided in the IP core network. Otherwise, the receiver (i.e. at the RTP-level or audio/video decoder) will be in charge of it.

Some considerations about the impact of jitter on the QoS perceived by the end-user in the OPTIMIX architecture will be reported in D3.2c [11], being the same as for the introduction of other mechanisms in the IP wired core network (i.e. the designed Traffic Engineering system with fault resilience capability, also modelled in the simulation chain).

In the carried out analysis we have used the same traffic aggregate as in the comparison between AWTP and DWFQ, described in Sec. 3.1.1.2.

3.1.2.1 Feedback delayed

We start evaluating the impact of the feedback delay: as a consequence the information about the end-to-end delay at the receiver arrives after some time to the ingress border router and hence there will be a delay in the router reaction and decision in either promoting or downgrading the traffic.

We have considered three different values of the feedback delay: 20, 50 and 100 msec. for each value we have analysed how quick and stable is the system in promoting and downgrading the UDP traffic, as well as the 99th percentile of the end-to-end delay of the UDP traffic allowed to change class and its jitter.

Considering the shortest delay, we can notice that the behaviour of the system is quite the same as in the case of no delay: the graphical representations of the DSCP value along the simulation time (Figure 29) are very similar, therefore the introduced delay does not impact significantly the speed of the reaction and the stability of the system. As a consequence also the 99th percentile of the end-to-end delay of the jumping UDP traffic is very similar to the one without latency in the feedback transmission (Figure 30).

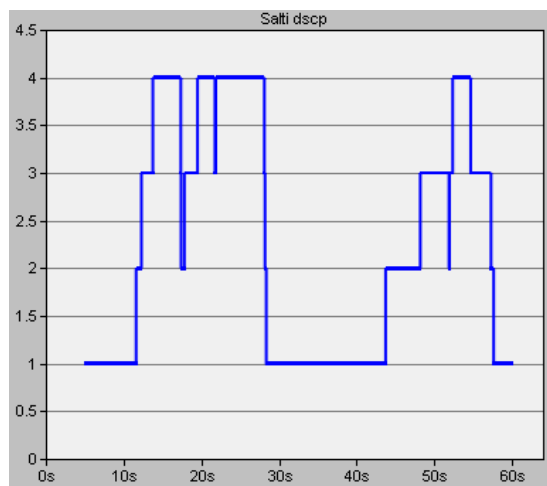


Figure 29 - AWTP, promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$, feedback delay of 20 msec.

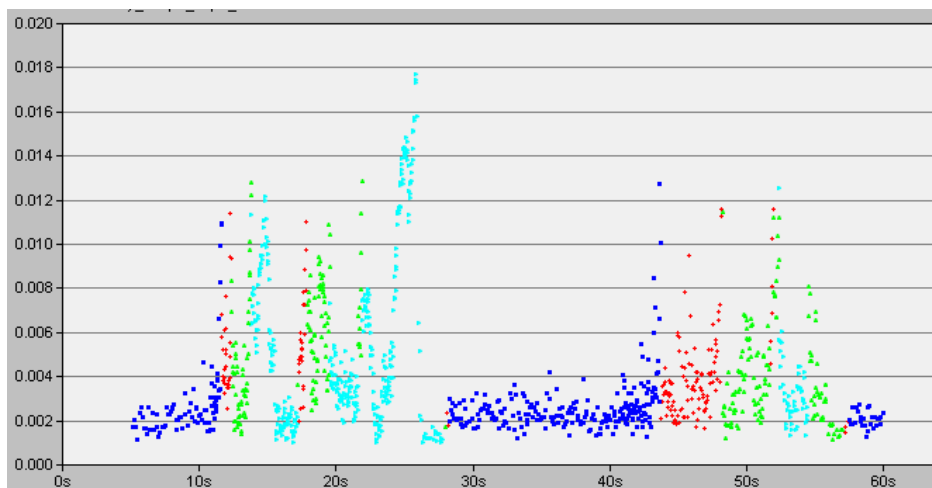


Figure 30 - AWTP, 99th percentile of the end-to-end delay of the UDP traffic with a feedback delay of 20 msec.

In the next case the feedback delay is 50 msec: the DSCP values (Figure 31) are quite similar to the previous case during the first traffic change, while they vary significantly during the second traffic change. When the traffic increases for the first time the system reacts quite fast with only 2 seconds of delay from the traffic change, like in the previous case and the case without latency in the feedback transmission. Instead, when the traffic increases for the second time, the delay in the reaction is quite high (10 seconds). If we move to the case of 100 msec of feedback delay (Figure 32),

we can see the same behaviour, but this time the delay in the reaction at the second traffic change is lower than in the case of 50 msec of feedback delay (5 seconds).

Comparing the three analysed cases we can underline that the overall system has very similar behaviour during the first traffic change, while the performance is different at the second traffic change. This is due to the fact that the introduced delay affects enough the system behaviour in a way that the delay samples of the jumping traffic are still quite different when the second traffic change appears, leading to significantly diverse reaction time and performance in the end. Concluding that a feedback delay of 50 msec is the worst case while higher feedback delay are better is wrong, because the impact of the feedback delay on the traffic rearrangements at the first routers strongly depends on the source traffic profile and therefore extremely variable in general. Therefore, we can state that a short latency in the feedback transmission makes the system more robust and performing whatever the addressed scenario.

Moreover we have to take into account that the feedbacks traffic is control traffic and, according to best practices of network configuration, it is delivered as traffic with a high priority and consequently cannot experience high delays in practice, i.e. a feedback delay of 20 msec is a realistic hypothesis even for large core networks.

Figure 33 and Figure 34 depict the behaviour of the 99th percentile of the delay of the jumping UDP traffic respectively in the cases of 50 and 100 msec of feedback delay. Both show a behaviour that is in accordance with the DSCP changes as illustrated in the previous figures (Figure 31 and Figure 32 respectively).

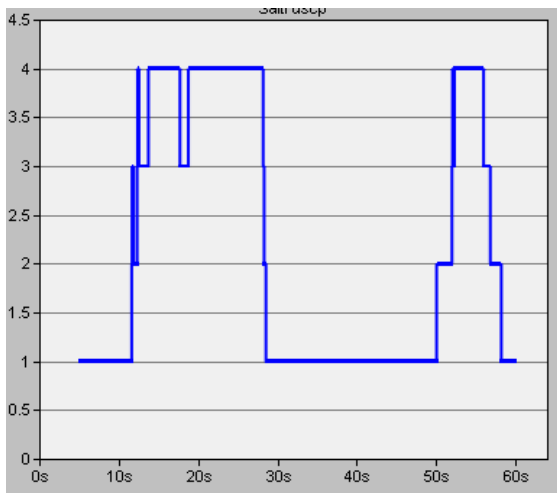


Figure 31 - AWTP, promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$, feedback delay of 50 msec.

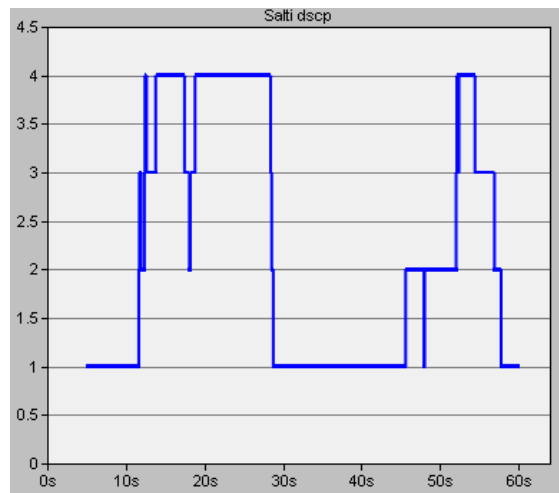


Figure 32 - AWTP, promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$, feedback delay of 1000 msec.

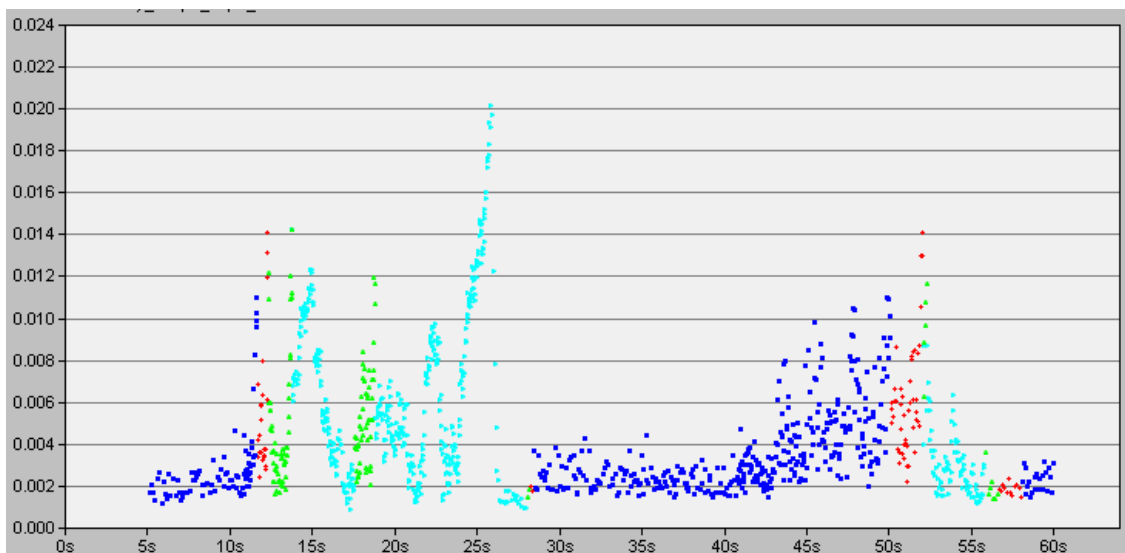


Figure 33 - AWTP, 99th percentile of the end-to-end delay of the UDP traffic with a feedback delay of 50 msec.

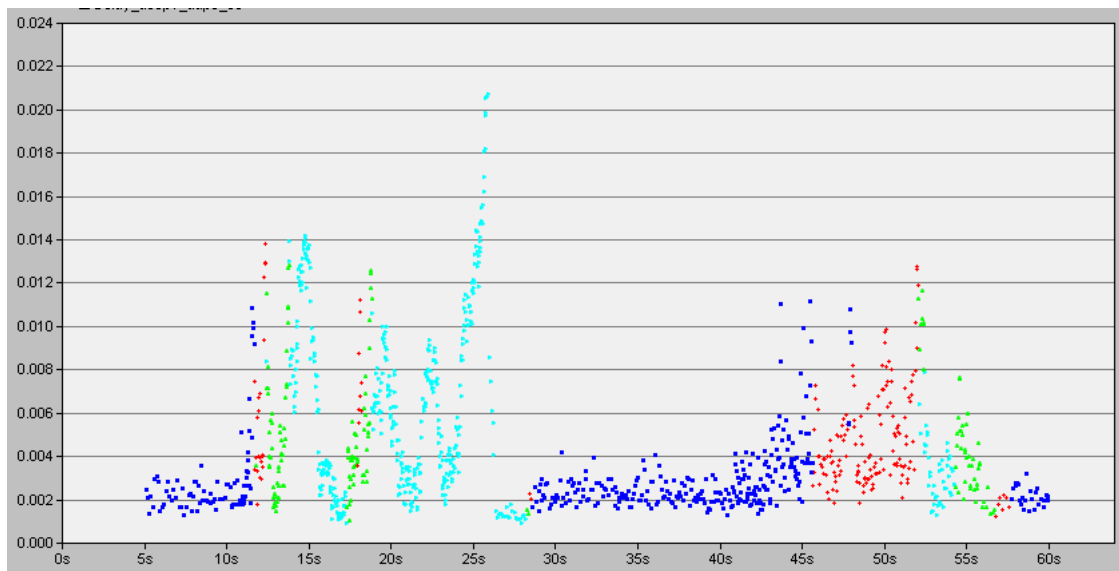


Figure 34 - AWTP, 99th percentile of the end-to-end delay of the UDP traffic with a feedback delay of 100 msec.

Let's now consider the effect of the feedback delay on the end-to-end QoS. In order to do this, we have also evaluated the end-to-end delay and the jitter. In the case of a feedback delay of 20 msec the end-to-end delay is depicted in Figure 35, which shows peaks in correspondence of the traffic changes, in particular the first one, where the congestion is very high and the connection is near the saturation. The delay in this case has a variance of 9.23 msec^2 , a 99th percentile of 10.449 msec and finally the peak to peak distance of 17.488 msec.

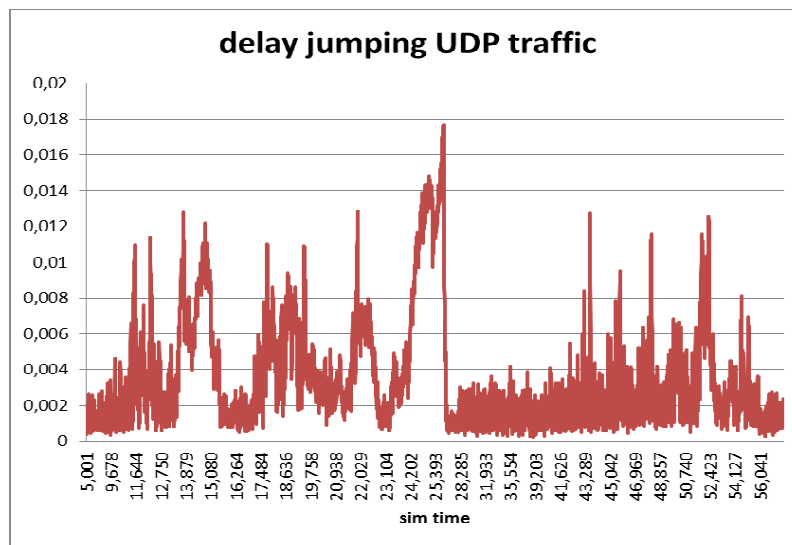


Figure 35 - End-to-end delay of the UDP traffic that can change class, in the case of 20 msec of feedback delay.

If we now consider the case of a feedback delay of 50 msec, in Figure 36 we can notice that the end-to-end delay has higher peaks and in general different values, because the UDP traffic that can change class has been managed differently from before. In this case the delay variance is 10.8 msec^2 , the 99th percentile is 11.158 msec and the peak-to-peak distance is 19.981 msec.

Lastly, in the case of 100 msec of feedback delay the end-to-end delay (Figure 37) has the same behaviour as in the previous case, with slightly higher peaks. The delay variance is 11.875 msec^2 , the 99th percentile is 11.64 msec and the peak-to-peak distance is 20.609 msec.

Comparing the three analysed cases we can notice that an increase in the feedback delay causes a slight worsening in the end-to-end delay and also the jitter becomes larger, as reflected in the delay variance and the peak-to-peak distance. This means that the higher the feedback delay, the worse the user perception of the displayed video stream at the receiver.

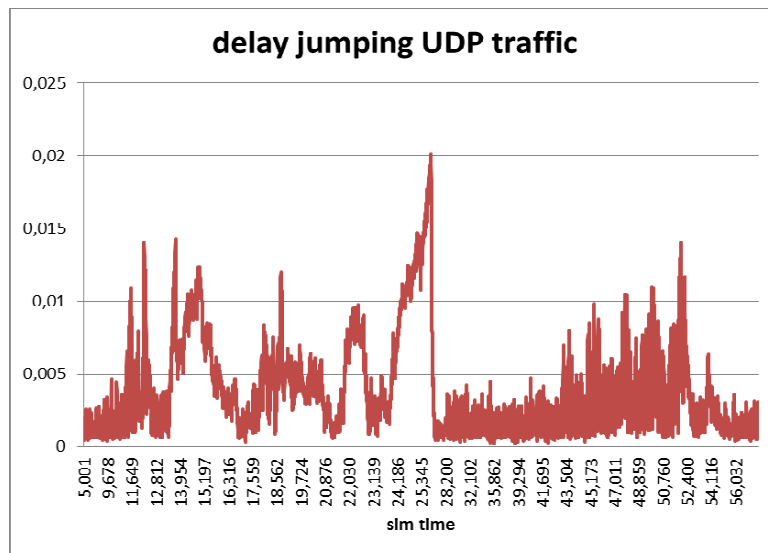


Figure 36 - End-to-end delay of the UDP traffic that can change class, in case of 50 msec of feedback delay.

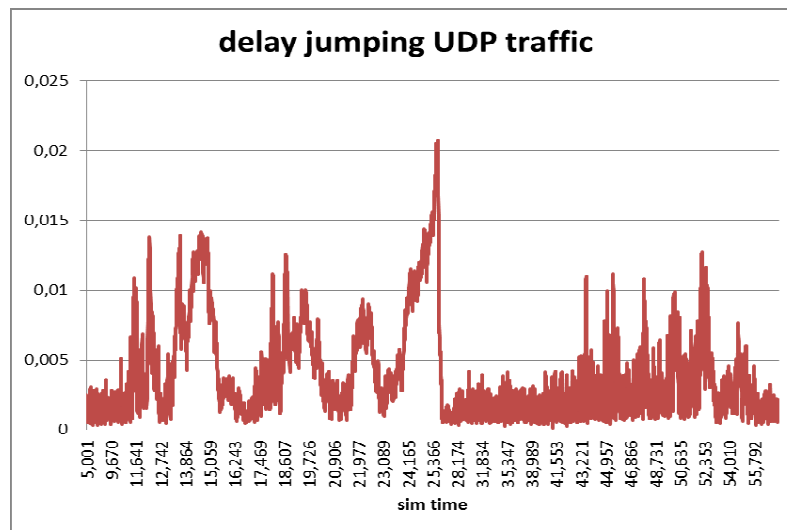


Figure 37 - End-to-end delay of the UDP traffic that can change class, in case of 100 msec of feedback delay.

3.1.2.2 Different number of hops to cross

In order to evaluate the designed solution for the absolute QoS support in a more realistic scenario, we have also to consider the case of more than three nodes to be crossed. For this reason we have made simulations using a network scenario in which there are 6 routers, as depicted in Figure 38. In this scenario, we expect to have a greater end-to-end delay, hence we have increased the threshold delay from 10 msec to 20 msec, as specified in the issued SLA.

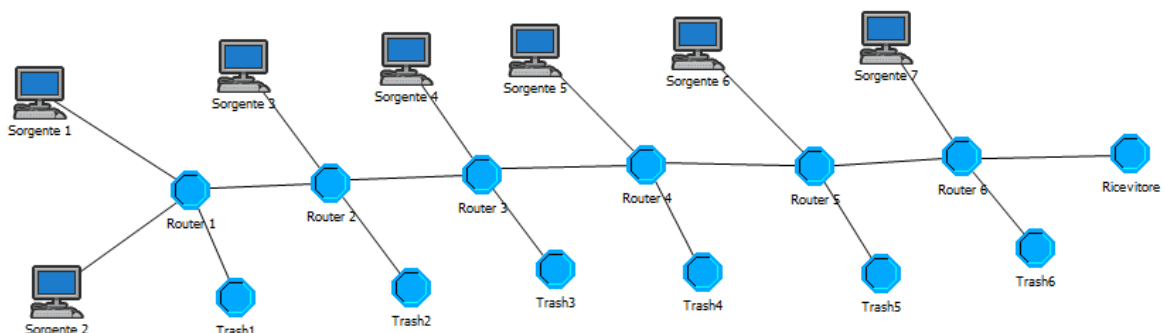


Figure 38 - Scenario with 6 nodes

The DSCP value (Figure 39) changes roughly as in the previously analyzed cases, when the traffic load increases. In particular we can notice that the system reacts quite fast to the traffic increases, both the first and the second one, while it is more conservative and stable when the traffic decreases. In fact, when the traffic decreases the first time (at around 25 sec) the system needs 6 seconds for coming back to the first class. The same behaviour can be observed when the traffic decreases the second time (at around 55 seconds). This is probably due to the fact that with a greater number of nodes the delay at the receiver is greater and the system requires more time to update all the delay values of the threshold buffer, which has 20 samples, than in the case of three nodes.

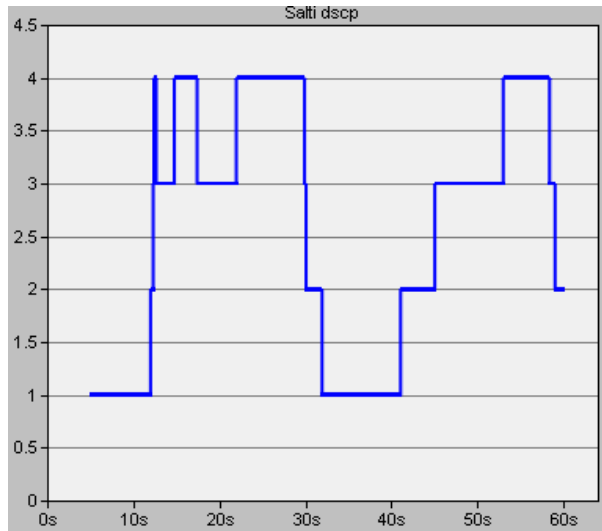


Figure 39 - AWTP, promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$, feedback delay of 20 msec, and 6 nodes

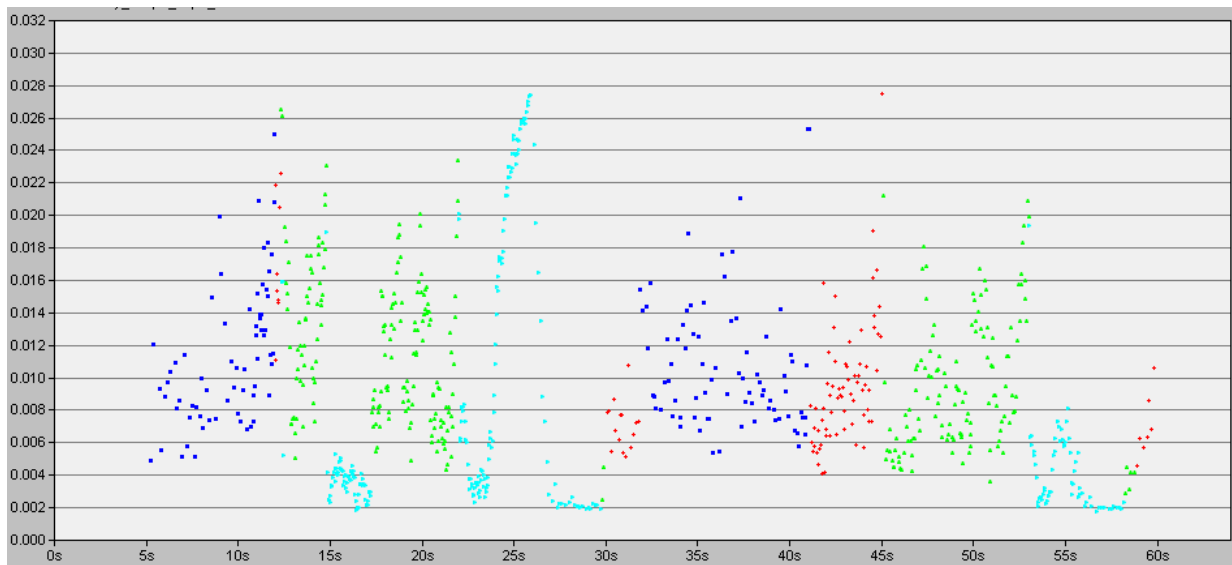


Figure 40 - AWTP, 99th percentile of the end-to-end delay of the UDP traffic with a feedback delay of 20 msec and 6 nodes

The expected increase in the delay is evident looking at the delay of the jumping UDP traffic (both the end-to-end simple values as shown in Figure 41 and 99th percentile as depicted in Figure 40), which has peaks of 27 msec instead of 18-20 msec as in the case of three nodes.

Looking at the jitter, we have that the delay variance is 32.7msec^2 , the 99th percentile is 17 msec and the peak-to-peak distance is 26.9 msec. These results show that when the number of nodes increases, not only the end-to-end delay is higher, but also the jitter increases. As a consequence we expect that the user perceived quality is worse than in the previously analysed cases.

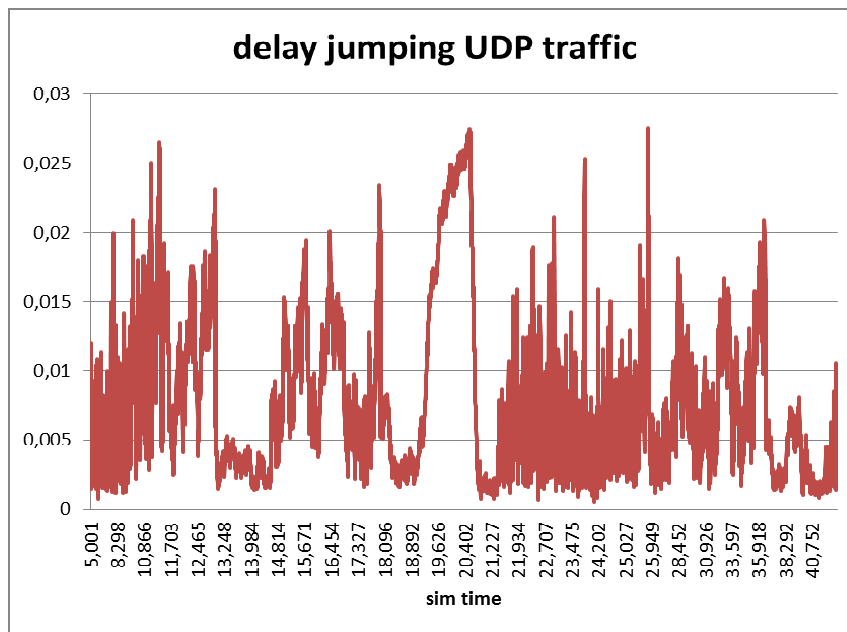


Figure 41 - End-to-end delay of the UDP traffic that can change class, in the case of 20 msec of feedback delay and 6 nodes

3.1.2.3 Different values for the quality factors of the scheduler

Finally, we have augmented the distance between the service classes in terms of provided relative QoS guarantees changing the quality factors: we have now configured them to 1, 4, 8 and 16, respectively for the four classes. In this analysis we have supposed to have a feedback delay of 20 msec.

Looking at the DSCP value picture (Figure 42), we can notice that the system reacts quickly to the first traffic change, showing the same behaviour as in the previous case with quality parameters equal to 1, 2, 3 and 4, respectively for the four classes. Considering instead the second traffic change, the system is slightly slower, because it requires 5 seconds to react. This is mainly due to the fact that the traffic is moved differently into the classes and the delay of the UDP traffic increases more slowly (Figure 44). However, broadly speaking, the DSCP value trend is similar to the one tracked in the previously analysed cases.

The performance of the system shows a general improvement in the end-to-end delay of the jumping UDP traffic, highlighted both in the graphs of the end-to-end delay (Figure 44) and of the 99th percentile of the end-to-end delay (Figure 43); also the delay peaks are lower than in the previously analysed cases (about 14 msec instead of 18/20 msec). Considering the jitter, the delay variance is 4.17 msec², the 99th percentile is 7.34 msec and the peak-to-peak distance is 14.428 msec. It is clear that increasing the relative distance between the service classes in terms of provided QoS guarantees has caused also an improvement in the resulting jitter, since the variance, the 99th percentile and the peak-to-peak distance are lower than before. As a consequence, we expect a significantly improvement in the user perceived QoS at the receiver.

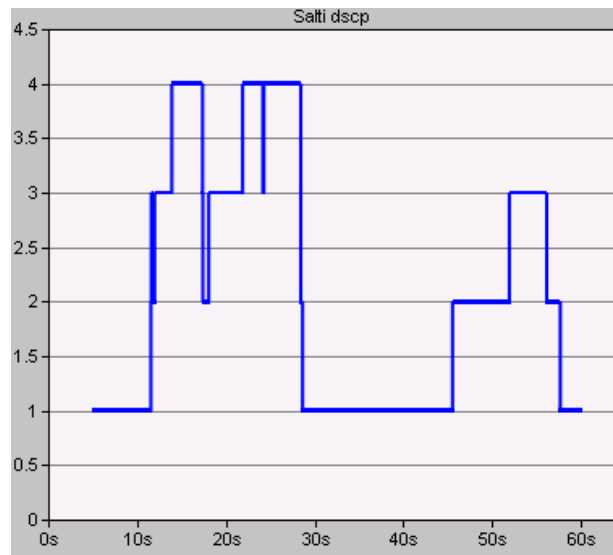


Figure 42 - AWTP, promotion by moving average with $K_{up}=2$, downgrading by threshold with $K_{down}=20$, feedback delay of 20 msec and quality factors 1, 4, 8, 16

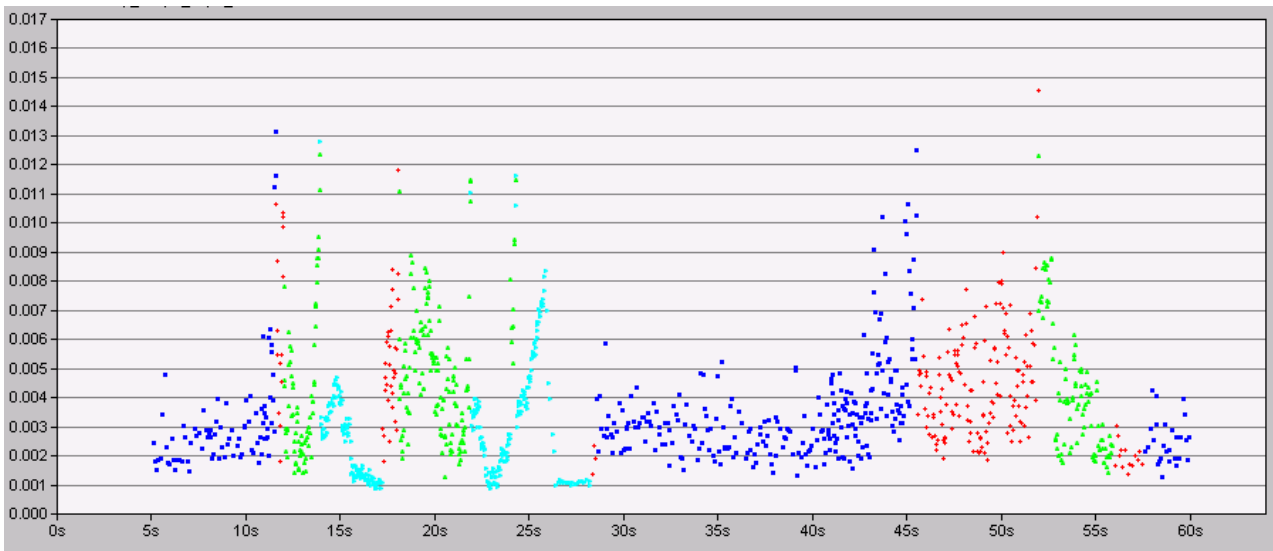


Figure 43 -AWTP, 99th percentile of the end-to-end delay of the UDP traffic with a feedback delay of 20 msec and quality parameters 1, 4, 8, 16

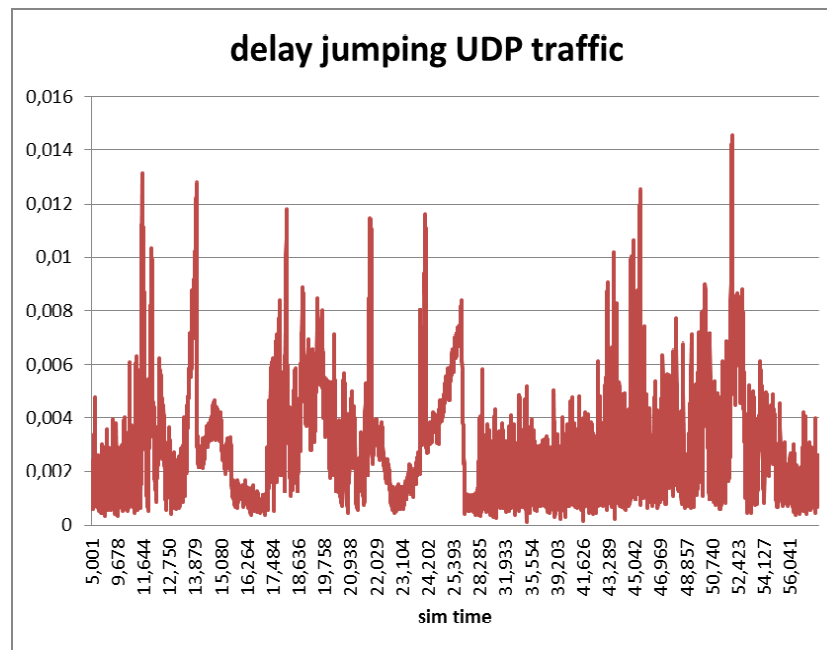


Figure 44 - End-to-end delay of the UDP traffic that can change class, in case of 20 msec of feedback delay and quality parameters 1, 4, 816

3.2 Impact of the designed traffic engineering mechanism with fault resilience capability

The goal of this section is to evaluate the deployment of the Traffic Engineering mechanism with fault resilience capability designed within the framework of the OPTIMIX project (see D3.1a [6] and D3.1b [7]) in an IPv6 network. The analysis is carried out in general as for a Next-Generation Network (NGN), but the primary goal is to understand how the OPTIMIX's JSCC/D system behaves in presence of a link or node failure in the network, when the designed system for robustness is implemented.

Such an analysis is also helpful to properly model the IP Network module in the simulation chain, which should consistently consider all the relevant components, mechanisms and functionality of a NGN.

The effect of the TE system is manifold. At a first glance, we could focus our attention on delay, delay jitter, loss and possibly re-ordering phenomena in case of a network fault.

The specification of the order of magnitude and nature of the mentioned impacts should be the result of considerations regarding the functioning of the involved protocols) i.e. MPLS (and mechanisms (e.g. FRR), as well as of the quantitative results collected by the already performed simulation analysis (see deliverables D3.1a [6] and D3.1b [7]). Last but not least, the specific application and network scenarios of OPTIMIX project is to be taken into account.

In order to provide reliable and consistent qualitative and quantitative conclusions, it is critical to well identify all the steps taken by the TE system when a link or node failure happens within the IP network. Both the operating of designed advanced Haskin and ONE-to-One Backup FRR techniques of MPLS and the process flow for the re-calculation of Working and Recovery Paths (WPs and RPs, respectively) are to be considered in detail. All the explanation and descriptions about these aspects are included in D3.1a ([6]) and D3.1b ([7]). Therefore, just the main results and issues are going to be reported hereafter also for the sake of clarity.

First, given a network (in terms of topology, link capacity and cost) and a set of unicast or multicast traffic demands, a set of WPS and related RPs is calculated for the accepted requests.

Second, MPLS signalling (by means of RSVP_TE or CRLDP) is used to establish the calculated primary and backup paths in the network.

Both these two operations are performed off-line and therefore do not have really an impact on the run-time operating and performance of the network.

In absence of faults, the loss and delay of the IP packets flowing from the ingress to the egress border routers only depend on the number of crossed nodes and the queuing process at each router interface.

When a failure happens, the enabled FRR techniques are activated. The affected traffic is first bounced back at the POD (Point of Detection) on the WP in the reverse direction and then switched into the associated RP at the PSL (Protection Switching LSR). By the RP the traffic reaches the egress border router either returning on the WP downlink the failure or not, depending on the issued FRR techniques (i.e. local or global repair, respectively).

Re-ordering phenomena can be avoided by applying buffering at the routers of the wp between the PSL and POD, therefore the considered traffic is affected only by loss and delay/delay jitter.

By FRR, loss is related to the packets that are sent to the impaired link or node before detecting the failure. Typically, the detection is performed at data-link layer within few ms, e.g. 10 ms for Ethernet. Of course, the actual number of lost packets is directly proportional to the packet rate of the issued multimedia flow.

The extra- delay and delay jitter due to the failure are related to the number of nodes in the WP segment between the POD and PSL, and the difference between the RP and WP lengths, as well as the crossing time of each IP router interface.

The first contribution can be made lower than 50 ms, as in SONET/SDH network, while the last one is strictly related to the class of service of the considered traffic and network load. An approximation of 10 ms was made during our previous analysis, as a reasonable value for multimedia data in a NGN.

Depending on the size of the network and the type of optimization carried out when calculating the WPs and RPs for the accepted traffic demands (i.e. of either the spare capacity or the overall switching delay in the network), the difference between the lengths of the primary and backup paths can be of a few nodes and the number of links between the POD and PSL could be 3 at most. Therefore, the extra delay experience by the traffic concerned by a network failure should be lower than 100 ms on the whole, even for big metropolitan/geographic telecommunication infrastructures.

In the end, the extra-delay/delay jitter experienced by the traffic in case of failure is of some tens of ms up to 100 ms, with a very high probability. An interesting point to further investigate could be the new statistical delay distribution of the multimedia data, which is not so easy to determine in general. However, it is worthwhile to highlight that due to the buffering applied at the routers between the POD and PSL, a burst of packets is expected at the receiver, at least as long as the said buffers are being emptied. After that, the received packets should experience a delay/delay jitter along the RP that is of the same nature as the one along the WP before the failure, apart from a possible different number of nodes to be crossed that can lead to a longer or shorter end-to-end delay within the IP network.

A simple but consistent modelling of the IP network for a link or node failure managed by the designed TE system could be as follows:

- 1- to discard packets for a period equals to the detection time of a network or node failure (e.g. 10 ms)
- 2- during the traffic switching period, to buffer packets for an extra-interval of time randomly selected from 20 to 100 ms (for example with a uniform distribution), other than the delay applied in ordinary network conditions
- 3- to apply loss and delay to the following packets of the issued multimedia flow, as in ordinary network conditions (i.e. without a failure), possibly with a different average value (depending on a difference in the lengths of the wp and associated rp)
- 4- to inject an additional delay or short burst when the issued multimedia flow comes back to the WP (after the fault recovery). Because the WP and RP could have a different length and a buffering could be required at the PML (Protection merging LSR) to avoid re-ordering phenomena when the primary path is shorter.

Concerning the operating of an ordinary network, as well as the specific scenarios of OPTIMIX project, the analysis should also consider the addition of new traffic demands, changing of transmission capacity for core wireless links, as well as the user mobility, which all require the calculation of further couples of WPs and RPs run-time. For the former, we do not have really an issue, because the new traffic will be admitted within the network only when the related primary and backup paths have been established. For the latter ones, a new couple should be calculated and configured transparently to the user, who can move from one base-station/access point to another, while in streaming or involved in interactive audio/video communications.

From simulation results performed by a laptop PC equipped with an Intel Core Duo T2250 1.73 GHz of 2 GB RAM, and GLPSOL version 4.9 (the solver LP/MIP GPLK standalone [8]) for the unicast case (see D3.1a [6]) and by a PC laptop equipped with Intel Core 2 Duo E8400 3 GHz with 3.23 GB of RAM, and version 4.9 of GLPSOL for the multicast case (see D3.1b [7]), the processing time of the designed TE system can be of some hundreds of ms to some tens of s, depending on the point-to-point or point-to-multipoint nature of the new WPs and RPs to be calculated and their complexity. While, the used RAM is not really an issue.

However, if we consider just a few couples of primary and backup paths, either with Haskin or One-to-One Backup FRR, with either fixed or variable length U_s , the said time is of some hundreds of ms (as reported in the last results of D3.1b [7]), even without any optimization of the employed ILP solver. Such an optimization, which should avoid at least the re-building of all the constraints in the ILP system at any new run-time calculation, the processing time should significantly reduce to less than 100 ms. Regarding the WPs and RPs setup, even for large network no more than some tens of ms of delay are introduced.

On the whole, the resulting latency should allow for seamless user mobility, still maintaining robustness to network faults.

For this last issue, an improvement could be the periodic pre-calculation of WP and RP pairs for those traffic demands that could require it in future. For example, couples of WPs and RPs can be identified before hand for every base-stations/access points close to the ones the users are currently connected, and if necessary, resources pre-allocated for them. In a real implementation, the integration between the designed TE systems with the control plane proposed in OPTIMIX can be really profitable for the purpose. Indeed, the triggering engine makes available feedbacks about the wireless connections to both the mobility management components and protocol, as well as to the TE system, allowing them to work synergically and well synchronized.

In terms of modelling of the IP Network module, the calculation of a new couple of WP and RP for an already active multimedia flow can be managed the same way as the switching of the related traffic from an RP to the associated WP after a failure recovery, since in this case the switching from an old WP to the new wp happens and the process is quite similar. Essentially, the introduction of a short extra-delay on the multimedia data, when the new WP is longer than the old WP. Otherwise, a buffering to avoid re-ordering phenomena is applied at the egress of the network.

The present analysis is critical in order to evaluate the quality of the received multimedia data in case of network faults. The additional loss and delay/delay jitter on the transmitted packets have an impact. The RTP operations, loss concealment and resilience mechanisms at transport and application layers should compensate enough for them in order to have an acceptable quality in all network conditions.

Accordingly, the IP Network module of the simulation chain will be enhanced. The more relevant effects of a network fault when deploying the designed TE system will be properly modeled and an analysis on the resulting QoS will be carried out.

4 Security

In the previous deliverables D3.1a [6] and D3.1b [7] we have already introduced the enhancements that were analyzed during the OPTIMIX project.

In D3.1a we have published the analysis of progressive encryption. This method can spare CPU resources during the ciphering by running the encryption algorithm on concatenated packets without costly reinitialization. The overhead is that we have to signal the offset of the packets (the offset is calculated from the last initialization) in each packet. In that analysis we have investigated the AES cipher and the HMAC-SHA1 authentication algorithm.

In D3.1b we have published the analysis of a new cipher that can outperform the commonly used AES-ICM method. This cipher is the ChaCha cipher, a result of the ECRYPT I-II EU projects from D. J. Bernstein. Also, in this document, we have introduced two novel integrity protection mechanisms. One of them is an approximate authentication algorithm and a second one is a method to create sections in the packet and apply integrity code for all of them individually.

The SRTP module has been integrated in both in the simulation chain and the test bed. In this document we publish the final results of the investigations.

4.1 SRTP module

In the SRTP module ChaCha cipher is integrated in the simulator, where we also implemented an enhanced version of the APACE approximate authentication algorithm. The original APACE algorithm has been implemented for the PHOENIX project. In the simulation using the init file we can set:

- No ciphering, no authentication (disable SRTP)
- Two different ciphering methods: AES-ICM or ChaCha (independently of the authentication)
- Two different authentication methods: HMAC-SHA1 or APACE (independently of the cipher)

We have not implemented the progressive encryption in the final SRTP module. The reason is that in the case of multicast WiFi situation, the same frame can be sent several times (see D3.3c [12] for details). In this case we have to reset the cipher back to its previous state, since the repeated frames have the same offset. This reinitialization has a bad impact on the performance. We are working on the solution, preserving the good performance, we have observed without multiple retransmissions.

Also, the second integrity protection mechanism, which creates sections in a frame, has not been implemented. The reason is that from measurements with real WiFi transmissions we deduced that this method cannot significantly improve the performance. These measurements are presented in D3.3c [12] and the deduction to the integrity protection mechanism is presented below.

4.2 Performance of the ChaCha cipher

We have analysed the performance of the ChaCha cipher comparing it to the widely used AES-ICM method. During the analysis we created 1024 byte long data arrays and ciphered them with the two algorithms and measuring the execution time. Both ciphers took a complete initialization and encryption phases. We repeated the tests 10.000 and 100.000 times and calculated an average throughput value from the results. We have performed the tests on two very different architectures: first, a strong (server like) desktop PC, with an Intel Q6600 quad-core processor; second, an embedded device, used in Access Points, with a 266 MHz Broadcom MIPS processor.

Repeating the tests 10.000 and 100.000 times made no significant difference in the results, so the calculated average throughput is considered to be independent of other mechanisms during the test. The test results can be seen in Figure 45.

As the test result shows, the ChaCha algorithm is truly faster than the AES-ICM cipher. This result was expected. Moreover, we can see that in case of a strong CPU, ChaCha outperforms almost 4 times the AES algorithm, while in the case of a weak CPU, the ChaCha performance gain is “only” doubling the speed of AES-ICM. The AES cipher we used in the test was a software algorithm, but it used a large set of precalculated tables. The ChaCha was implemented with C language, without any assembly code. The test has also proven that the SRTP module with ChaCha can be implemented even on embedded devices or on mobile phones.

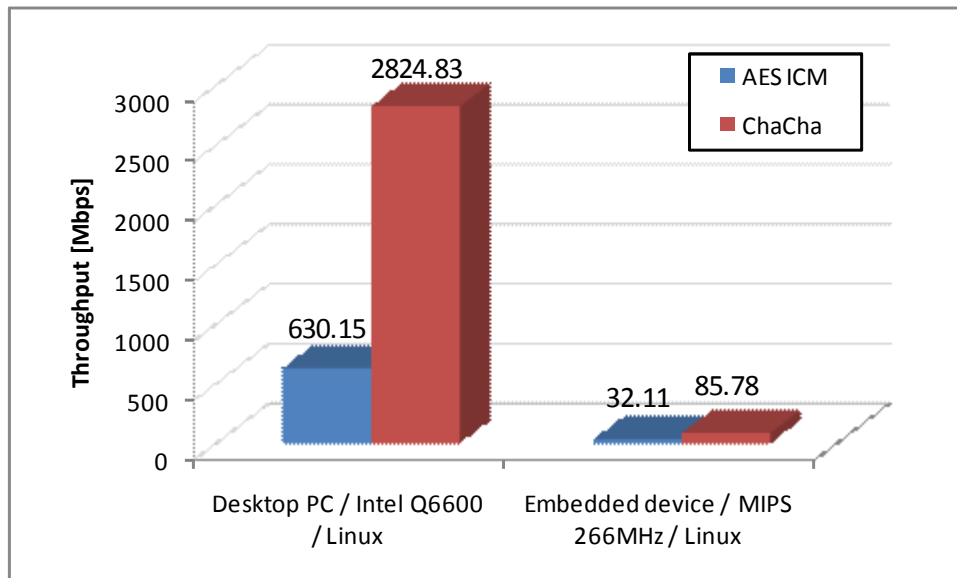


Figure 45 – Cipher throughputs

4.3 Enhanced APACE algorithm

The APACE algorithm was first published in 2006. It is an approximate authentication algorithm designed with a dual purpose: first, to give approximation for the amount of bit error during transmission; second, to protect content from malicious modifications. The two purposes are dealt by the same mechanism, as we estimate the number of changed bits during the transmission. We consider that a small number of bit errors could be natural or harmless, while a large number of bit errors could be already malicious or natural, but also an indication of a heavily damaged frame. In our applications we will tolerate frames with small number of bit error and drop frames with high number of bit errors.

The original APACE algorithm uses a block of counters that count the number of 1s and 0s in the payload and increase the counters pseudo randomly. The authentication code is the value of the counter that can be recalculated and compared at the transmission endpoint. The difference of the in frame and calculated counter blocks gives an estimation of the bit errors. Previously we used 32 pieces of 2 bit wide counters. According to simulations, this was an optimal value for 188 byte long MPEG TS frame payloads.

In the current project, we made enhancements on the previous APACE version.

4.3.1 Counter block

The counter block size was changed to 8 pieces of 8 bit wide counters. First, this is optimized to a transport, where 1024 byte long packets are transferred. Second, some other modifications let us to increase the size of the counters (in the previous version it was a security issue). Optionally, the authentication code (the values of the counters) can be encrypted with a block cipher. This encryption prevents attacks on the counter, as manipulating the encrypted counter makes huge changes in the decrypted values. However, this way, even a single natural bit error in the encrypted counter causes totally wrong estimation during the authentication.

4.3.2 Counter settings

Originally, each message bit was associated to a counter using a pseudo random number generator. When the bit was 1, the counter increased. In the enhanced version, each message bit is associated to a counter, however, this time the association is based on a pseudo random number generator that has an uneven distribution depending on an initialization value. This is a security mechanism against message related authentication code modifications. Also, in the enhanced version, not only 1s but 0s affects the counters as well, preventing chosen plaintext type attacks. 1s increase, while 0s decrease counter values.

We have analyzed the APACE algorithm using simulations. In the simulator we inserted a bit error (flipping a random message bit) into the message and redo the message authentication code (MAC) calculation. Figure 46 shows the difference of the MAC between the correct and erroneous message as a function of the inserted bit errors. We have repeated the tests several times, with similar results, thus showing that uneven and uniform distributions for counter selection perform the same, while using an uneven distribution is a more secure choice.

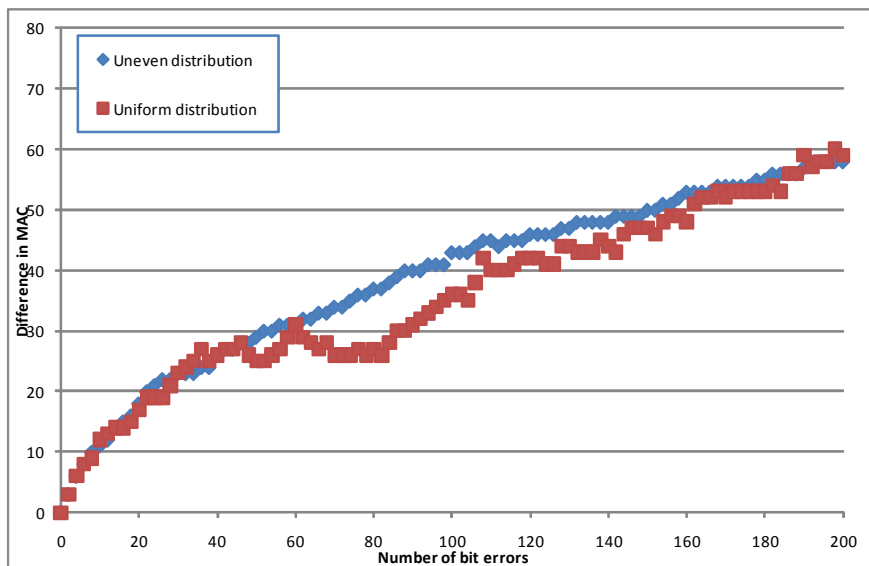


Figure 46 – Enhanced APACE algorithm

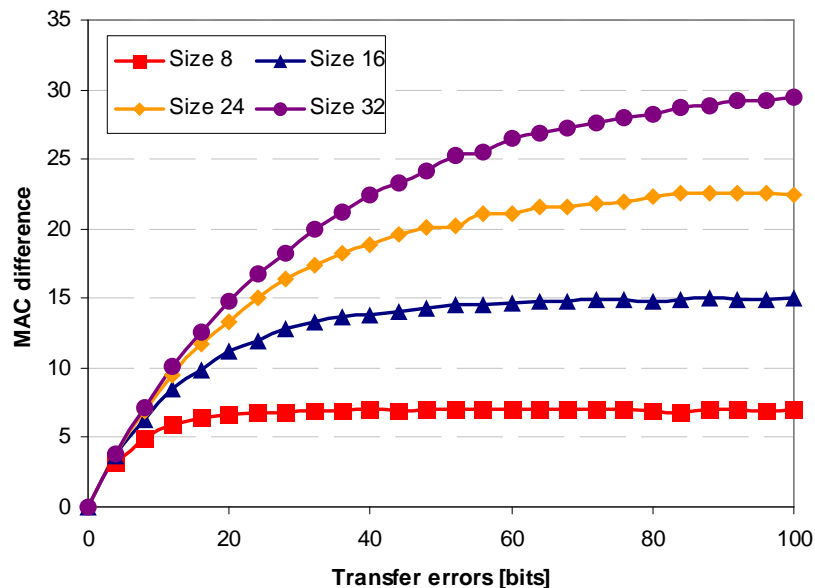


Figure 47 – The old version of the APACE algorithm

In Figure 47 we present the results of the same measurements in the case of the previous version of the APACE algorithm. The lengths of the counters were two bits only and the different colours represent different number of counters in the MAC block. The 32 x 2 bit counter is equal to the new 8 x 8 bit counter in terms of the MAC length. Comparing the results, the enhanced APACE has a better performance as the curve of the old version becomes flat after a certain number of erroneous bits, in contrast to the enhanced version. This means that the enhanced APACE can signal more error bits.

4.3.3 MAC difference calculation

We have also made tests altering the difference calculation used to estimate the number of bit errors. The original difference calculation is said to be linear, as we simply sum up the differences of the corresponding counters in the MAC block. Using wide counters, just like 8 bits wide, we can create non linear methods to estimate the errors. We have tested two methods:

- A step function. Until a certain threshold, the difference of a counter is counted as it is in the old version, however after this point there is a constant, larger than 1 factor multiplier applied.
- Square function. We calculated the difference of the counters and used the square of these values.

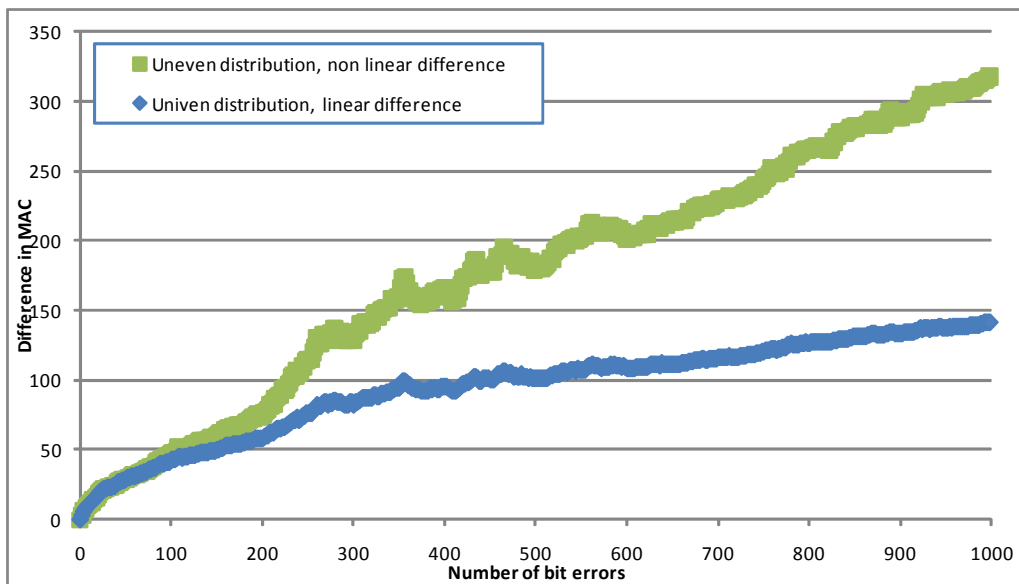


Figure 48 – Linear and non linear calculations

Figure 48 shows the calculated MAC difference curves in the function of the bit errors. The figure represents the results obtained using a step function for the non linear calculation. As it can be seen, the curve of the linear calculation method becomes more flat as the message bit errors grows, while the curve of the non linear method is steeper. When expecting large number of bit errors, the non linear calculation is better.

In the OPTIMIX project we used APACE also for security reasons. This means that we cannot allow too many bit errors, since in that case natural bit errors and malicious attacks on the message are indistinguishable. In the case of low number of bit errors, the linear and non linear calculations result almost the same, so we have selected the linear method.

4.4 Partial integrity

The method of the partial integrity protection is to create more sections within the payload message and create an individual message authentication code for a section. The MAC for the whole message is the block of the sections MACs. Here we use cryptographic hash code (e.g. HMAC-SHA1) for the MAC calculation. As a result, using this protection method, at the receiver side we can tell which sections were correctly transmitted and which were not. We do not have any estimation to the number of bit errors. If it is possible, the bad sections can be ignored, or they can be recovered from a repeated message.

In terms of CPU utilization this method is a very cheap error protection mechanism, as calculating a couple of message authentication codes are not so demanding. However, it is a considerable overhead in term of the bandwidth. First, the MACs take significant space in the message. The length of the HMAC-SHA1 is 160 bit, while the one used in SRTP is 80 bit. Second, the repetition of the whole message is bandwidth wasting. For this reason, this integrity protection method can be used in those situations where the CPU is an expensive resource and the bandwidth is cheap. Such environment could be the WiFi scenario, where Access Points are dumb, but they have lots of bandwidth comparing to a video call's bandwidth demand.

In the OPTIMIX project we have tested the performance of the partial integrity protection using 4 sections. In this imaginary scenario, there were a video call over the internet, and a wireless WiFi client was on the end. The WiFi Access Point repeated the frames on the WiFi side several times and applied the partial integrity code to them. In this test, we made real measurements on WiFi transmissions, changing also the speed of the transmitted frames. More details can be found in D3.3c [12].

In the measurements we have 3 cases:

Case 1: This is the most optimal solution using the best available transmission strategy (see 3.3c [12]). The repeated frames can have different transmission speeds. There is full integrity protection, the whole message is covered with one MAC.

Case 2: The repeated frames have the same transmission speeds. There is partial integrity protection with 4 sections.

Case 3: The repeated frames have the same transmission speeds. There is full integrity protection.

In the measurements we used 1000 bytes long frames and measured the time that is required to transfer the message with a gives average success rate (10%, 20%, ..., up to 100%)

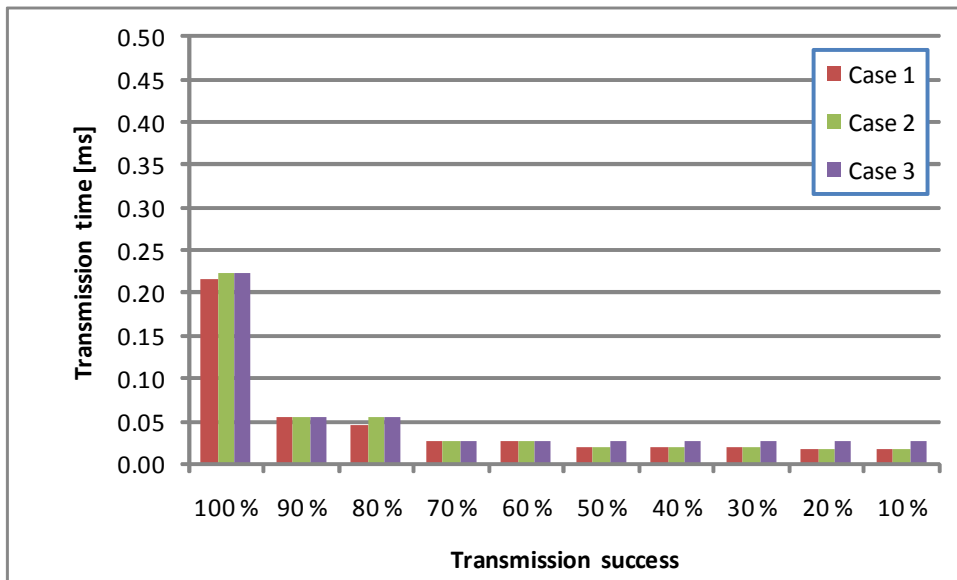


Figure 49 – Transmission in the 6m scenario

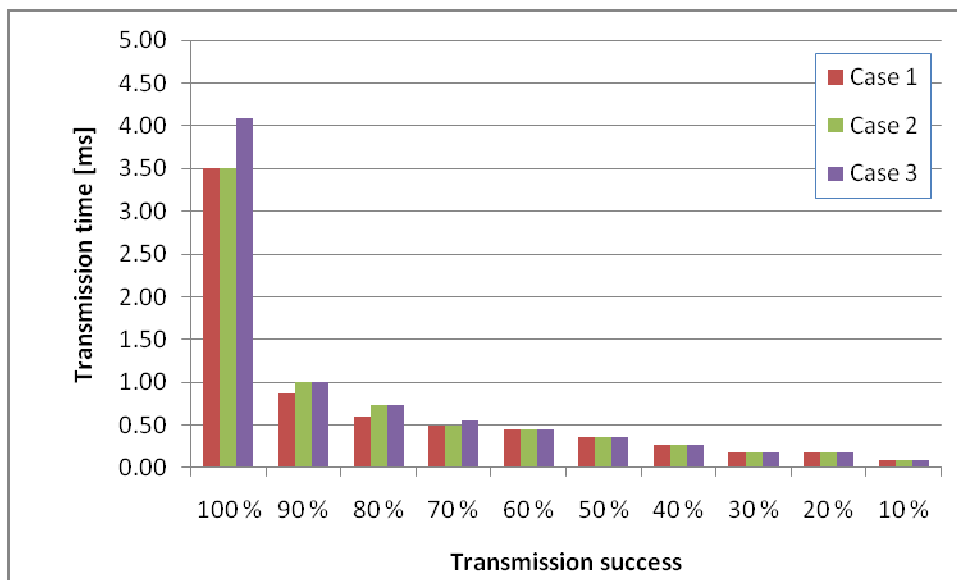


Figure 50 – Transmission in the 2 rooms scenario

In Figure 49 and Figure 50, we report the values obtained in two of the measured scenarios. First, we considered a distance of six meters between the Access Point and the receiver and, second, two rooms (i.e., three brick walls) between them. The results are different, however there is no significant difference among the cases. We got similar results in the other scenarios as well.

Table 4 – Transmission strategy for 6m

| Success | Case 1 | Case 2 | Case 3 |
|---------|---------------|--------|--------|
| 100 % | 7×36 and 1×48 | 8×36 | 8×36 |
| 90 % | 2×36 | 2×36 | 2×36 |
| 80 % | 1×36 and 1×54 | 2×36 | 2×36 |
| 70 % | 1×36 | 1×36 | 1×36 |
| 60 % | 1×36 | 1×36 | 1×36 |
| 50 % | 1×48 | 1×48 | 1×36 |
| 40 % | 1×48 | 1×48 | 1×36 |
| 30 % | 1×48 | 1×48 | 1×36 |
| 20 % | 1×54 | 1×54 | 1×36 |
| 10 % | 1×54 | 1×54 | 1×36 |

Table 5 – Transmission strategy for 2 rooms

| Success | Case 1 | Case 2 | Case 3 |
|----------------|---------------|---------------|---------------|
| 100 % | 7×2 | 7×2 | 45×11 |
| 90 % | 1×2 and 4×11 | 11×11 | 11×11 |
| 80 % | 1×2 and 1×11 | 8×11 | 8×11 |
| 70 % | 1×2 and 1×11 | 1×2 | 6×11 |
| 60 % | 5×11 | 5×11 | 5×11 |
| 50 % | 4×11 | 4×11 | 4×11 |
| 40 % | 3×11 | 3×11 | 3×11 |
| 30 % | 2×11 | 2×11 | 2×11 |
| 20 % | 2×11 | 2×11 | 2×11 |
| 10 % | 1×11 | 1×11 | 1×11 |

Based on our measurement experience, we can state that the use of the partial integrity protection has no significant advantage. Due to its CPU and bandwidth demands, we will not use this protection in the OPTIMIX project.

5 Conclusion

This document is the final reporting document of Task 3.1. It details the final results of transport and network protocol research and the final design of the corresponding mechanisms. The reader should also take a glimpse on D3.2c, which contains some more materials about transport and network protocols and open issues.

This deliverable shows a thorough overview on how DCCP transport protocol can be applied in a multicast environment. DCCP does not support multicast originally, but with tricky IP address changes and packet filtering multicast delivery can be implemented. Collecting and selectively filtering DCCP feedback packets by the aggregation server can be advantageous, because the server sending rate can be easily adapted to general needs. The overall packet loss ratio can be held under a given threshold by selecting the appropriate client from the multicast group: the methods are described here. We proposed different methods and algorithms to select the client at connection setup and during the communication. The number of clients in the multicast group can change: the change can be also handled using the introduced techniques.

In QoS issues, an analysis was made considering the AWTP and DWFQ scheduling algorithm, studying different approaches for promotion and downgrading the traffic: a fixed threshold, a moving average and a low pass filter. We compared the DWFQ and AWTP algorithms in the best own settings for absolute QoS provisioning, in order to choose the proper technique to be used in OPTIMIX architecture. Finally, we evaluated the impact of the best solution on the joint source-channel (de-)coding scheme proposed by the project.

Considering security, we have seen that several decisions could be taken, following the simulation results. The tests have proven that the SRTP module with ChaCha can be implemented even on embedded devices or on mobile phones. In the OPTIMIX project we used APACE also for security reasons. In the case of low number of bit errors, the linear and non linear calculations result in almost the same performance, so we have selected the linear method.

This deliverable closes the work of Task 3.1 without open issues.

References and glossary

5.1 References

- [1] Vida, R. & Costa, L., “Multicast Listener Discovery Version 2 (MLDv2) for Ipv6”, IETF RFC 3810, 2004
- [2] Kohler, Handley, Floyd, “Datagram Congestion Control Protocol”, Internet Engineering Task Force, RFC 4340, March 2006.
- [3] Sam Liang and David R. Cheriton, “TCP-SMO: Extending TCP to Support Medium-Scale Multicast Applications”, In IEEE Infocom, June 2002, New York.
- [4] Karl Jeacle and Jon Crowcroft, “TCP-XM: Unicast-enabled Reliable Multicast”, IEEE ICCCN, San Diego, October 2005.
- [5] S. Floyd, E. Kohler, and J. Padhye, “Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)”, RFC 4342, March 2006.
- [6] OPTIMIX project deliverable D3.1a, “Specification and preliminary design of transport and network layer protocols and mechanisms”. January 200.
- [7] OPTIMIX project deliverable D3.1b, “Specification and intermediate design of transport and network layer protocols and mechanisms”. January 2010
- [8] GLPK (GNU Linear Programming Kit) package. <http://www.gnu.org/software/glpk/>.
- [9] OPTIMIX project deliverable D3.2a, “Specification and preliminary design of network architecture”. January 2009
- [10] OPTIMIX project deliverable D3.2b, “Specification and intermediate design of network architecture”, January 2010
- [11] OPTIMIX project deliverable D3.2c, “Final design and functional/performance analysis of Network Architecture”, August 2010
- [12] OPTIMIX project deliverable D3.2c, “Final design and functional/performance analysis of Data Link Layer”, August 2010

5.2 Glossary

| | |
|-----------------|--|
| <i>AWTP</i> | <i>Advanced Waiting Time Priority</i> |
| <i>DCCP</i> | <i>Datagram Congestion Control Protocol</i> |
| <i>DiffServ</i> | <i>Differentiated Services</i> |
| <i>DP</i> | <i>Detour Path</i> |
| <i>FRR</i> | <i>Fast Re-Route</i> |
| <i>GLPK</i> | <i>GNU Linear Programming Kit</i> |
| <i>ILP</i> | <i>Integer Linear Programming</i> |
| <i>IP</i> | <i>Internet Protocol</i> |
| <i>ISP</i> | <i>Internet Service Provider</i> |
| <i>LER</i> | <i>Label Edge Router</i> |
| <i>LP</i> | <i>Linear Programming</i> |
| <i>LPF</i> | <i>Low Pass Filter</i> |
| <i>LSP</i> | <i>Label Switched Path</i> |
| <i>LSR</i> | <i>Label Switching Router</i> |
| <i>MA</i> | <i>Moving Average</i> |
| <i>MPLS</i> | <i>Multi Protocol Label Switching</i> |
| <i>NGN</i> | <i>Next-Generation Network</i> |
| <i>PDM</i> | <i>Proportional Differentiation Model</i> |
| <i>PLI</i> | <i>Programmazione Lineare Intera</i> |
| <i>PML</i> | <i>Path Merge LSR</i> |
| <i>POF</i> | <i>Point of Failure</i> |
| <i>POR</i> | <i>Point of Repair</i> |
| <i>PS</i> | <i>Protection Switching</i> |
| <i>PSL</i> | <i>Path Switch LSR</i> |
| <i>QoS</i> | <i>Quality of Service</i> |
| <i>QoS</i> | <i>Quality of Service</i> |
| <i>REA</i> | <i>Recursive Enumeration Algorithm</i> |
| <i>RP</i> | <i>Recovery Path</i> |
| <i>RR</i> | <i>Re-Routing</i> |
| <i>RT</i> | <i>Real Time</i> |
| <i>RTP</i> | <i>Real-Time Protocol</i> |
| <i>SLA</i> | <i>Service Level Agreement</i> |
| <i>TCP</i> | <i>Transport Control Protocol</i> |
| <i>TCP-SMO</i> | <i>TCP Singlesource Multicast Optimization</i> |
| <i>TE</i> | <i>Traffic Engineering</i> |
| <i>TFRC</i> | <i>TCP Friendly Rate Control</i> |
| <i>TLV</i> | <i>Type-Length-Value</i> |
| <i>UDP</i> | <i>User Datagram Protocol</i> |
| <i>WFQ</i> | <i>Weighted Fair Queuing</i> |
| <i>WP</i> | <i>Working Path</i> |
| <i>WTP</i> | <i>Waiting Time Priority</i> |