ONTIC: D5.3: ONTIC Subsystem Integration

Juliette Dromard, Philippe Owezarski, Miguel Angel López, Carlos Area, Miguel-Ángel Monjas, Alejandro Bascuñana, Martin Vicente, Fernando Arias, Alberto Mozo, Sandra Gómez, et al.

Online Network Traffic Characterization

ONTIC Subsystem Integration

ONTIC Project
(GA number 619633)

Deliverable D5.3
Dissemination Level: PUBLIC

Authors

Juliette Dromard, Philippe Owezarski
LAAS-CNRS

Miguel Angel López, Carlos Area
SATEC

Miguel-Ángel Monjas, Alejandro Bascuñana
Ericsson Spain

Vicente Martín, Fernando Arias
DellEMC

Alberto Mozo
Sandra Gómez
Bruno Ordozgoiti
UPM

Version

ONTIC_D5.3_2017.02.03_1.0.DOCX

Version History

Version	Modification date	Modified by	Summary
0.01	2016-10-30	Ericsson	Structure proposal
0.10	2016-11-18	CNRS, SATEC	Contribution to UC#1
0.20	2016-12-23	Ericsson, DellEMC	Contribution to UC#2
0.21	2017-01-17	Ericsson	Rework and heading sections
0.22	2017-01-18	Ericsson	Configuration and parametrization enhancement
0.3	2017-01-20	Ericsson, UPM	Restructuring of section 7.2.3. Creation of Annex A
0.4	2017-02-01	UPM	UC#3 contribution
0.5	2017-02-02	Ericsson	Ready for QA review
1.0	2017-02-03	SATEC	QA Review Final version

Quality Assurance:

Role	Name
Quality Assurance Manager	**Miguel Ángel López Peña (SATEC)**
Reviewer #1	**Alberto Mozo (UPM)**
Reviewer #3	**Panos Georgatsos (ADAPTit)**

Table of Contents

List of figures

List of tables

1 Acronyms

Acronym	Defined as
ARIMA	Autoregressive Integrated Moving Average
CSP	Communication Service Provider
DP	Data Preprocessing (unit)
EERCP	End-to-End Congestion Control Protocol
FC	Forecasting (unit)
FM	Forecasting Module
GUI	Graphic User Interface
ISP	Internet Service Provider
MMFA	Max-Min Fair Optimization Algorithm
ORUNADA	Online and Real-time Unsupervised Network Anomaly Detection Algorithm
PCA	Principal Component Analysis
PCAD	Proactive Congestion Avoidance Driver
PCAP	Packet Capture
QoE	Quality of Experience
RTT	Round-Trip Time
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
XML	Extensible Markup Language

Table 1: Acronyms

2 Purpose of the Document

Deliverable D5.3 purpose is to provide information about how the algorithms developed in scientific work packages have been applied in use cases. Although it is possible to assume that integration should have been straightforward, in general, adaptations, configurations, and transformations are needed. For instance, the following adaptation could have been needed:

- Interface adaptation: it means not only protocol (many times a specific protocol wrapper has been designed), but also data model adaptation. Sometimes it has been also interconnected using other off-the-self systems such as data brokers, cloud platforms…
- Redesign if the algorithm was designed in a language/technology different from the one used in the use case.

Also, the following information would be needed in order to fully understand how the algorithms are run:

- The parameters used in the algorithm implementation within the use case. For instance, if considering a Spark Streaming-based algorithm, the size of the windows or the thresholds used.
- The configuration parameters for the platform where the algorithm is run (RAM, processors…)

3 Scope

This document provides information about algorithm usage in the use case prototype implementation. However, generic details about use case implementation can be found in corresponding deliverables:

- ONTIC. “Deliverable D5.4. Use Case #1: Network Intrusion Detection” [4].
- ONTIC. “Deliverable D5.5. Use Case #2: Adaptive Quality of Experience Control” [5].
- ONTIC. “Deliverable D5.6. Use Case #3: Proactive Congestion Detection and Control [6].

4 Intended Audience

The intended document audience includes not only all the partners in the ONTIC consortium (especially those involved in gathering requirements, and in designing, implementing and validating the prototypes) or the receivers of the project. It also includes any reader interested in understanding the ONTIC use cases and the business principles that guide the research within the project.

5 Suggested Previous Readings

It is expected that a basic background on Information and Communications Technology (ICT) is sufficient to address the contents of this document; however, some previous readings are suggested (mind that in deliverables D5.1 and D5.2, use cases #2 and #3 were swapped):

- ONTIC. “Deliverable D5.1. Use Case Requirements” [2].
- ONTIC. “Deliverable D5.2. Use Case Requirements” [3].
- ONTIC. “Deliverable D5.4. Use Case #1: Network Intrusion Detection” [4].
- ONTIC. “Deliverable D5.5. Use Case #2: Adaptive Quality of Experience Control” [5].
- ONTIC. “Deliverable D5.6. Use Case #3: Proactive Congestion Detection and Control” [6].

6 Executive Summary

Three different use cases has been tested in ONTIC in order to determine the feasibility of the ONTIC algorithms.

Use case #1 deals with detecting anomalies in real-time in network traffic. This use case consists of two main components: a detection algorithm implementing unsupervised machine learning mechanisms, and a dash board for displaying network traffic characteristics in real time as well as raising the alarms when an anomaly is detected by the detection algorithm. Both components have been developed. This deliverable shows how they have been integrated in order to appear as a single system.

UC #2 has overcome the limitations of the ONTS regarding the use case requirements by creating a reduced labeled dataset made of network traffic traces. Labels are assigned according to the quality perceived by human viewers when watching videos. After appropriate cleaning and normalization, an adaptation of the FreeScan procedures are carried out: a clustering model is extracted from the dataset (considering both a training and a test subset); clusters are mapped against labels, which are turned into numeric values; finally, the model is applied to new network traffic traces, and a numeric quality index is associated to each of them; finally, an average value for time windows is computed for different locations. When the average value sustainable crosses a threshold, the QoE degradation is thus detected and appropriate alleviation measures are implemented.

Use Case #3 focus on congestion control. Nowadays, end to end host mechanisms for congestion control such as TCP are widely deployed, scale to existing traffic loads, and share network bandwidth applying a flow-based fairness. However, in a context where it is expected, in the short term, that more than 90 percent of the Internet traffic will go through data centers, TCP converge slowly to steady sending rates and cannot isolate datacenter tenants from interfering with each other.

UC #3 proposes a combined deployment of proactive congestion control protocols by explicitly computing sending rates and rate enforcement points placed in the edges of the network as a solution to establish the right way to mitigate these issues. The functionalities developed in this use case contain the following elements: (a) a scalable and distributed max-min fair optimization algorithm (MMFA) to compute max-min fair rates that iterates rapidly until converging to the optimal solution, (b) a proactive end-to-end congestion control protocol (EERCP) that deploys the former max-min algorithm to compute the session sending rates and enforces the fulfillment of these rates at the edges of the network and (c) a machine learning component that implements forecasting capabilities in order to provide the sources with accurate max-min fair assignments while the protocol iterates to convergence. Regarding the unfeasibility of a realistic deployment, UC#3 will be demonstrated by means of simulations run on top of a discrete events based simulator.

7 Subsystem Integration

7.1 Use case #1: Network Intrusion Detection

7.1.1 Use case description

ONTIC UC #1 designed a new autonomous anomaly detection system based on original unsupervised machine learning algorithms designed for that purpose. The most important feature of the anomaly detector is that it does not rely on previously acquired knowledge, it does not need any training phase or labeled data, and it is expected not to leverage in most of the cases on a human operator for making a decision on the status of detected anomalies (legitimate vs. attack or intrusion for instance). It aims also at triggering the appropriate counter-measures in most cases.

However, based on project research results in WP4, it appears that it would not be possible for the anomaly detection to autonomously make a decision for all anomalies. The new functionality that is required, and has been added in the design of the new anomaly detection system is a network traffic analytic dashboard. It aims at providing the human administrator with the required elements gained by the detection algorithms in order for her/him to decide whether the anomaly is legitimate or not, and apply the suited counter-measure.

7.1.2 Implementation details

This system has been designed specifically for the purpose of the ONTIC project, i.e. from scratch.

Figure 1 represents the high level proof-of-concept architecture showing the PCAP file, containing traffic traces, as the input to the two subsystems – the Anomaly Detection and the Network Traffic Dashboard subsystems. The results of the anomaly detection process, as XML files, are provided to the dashboard as well.

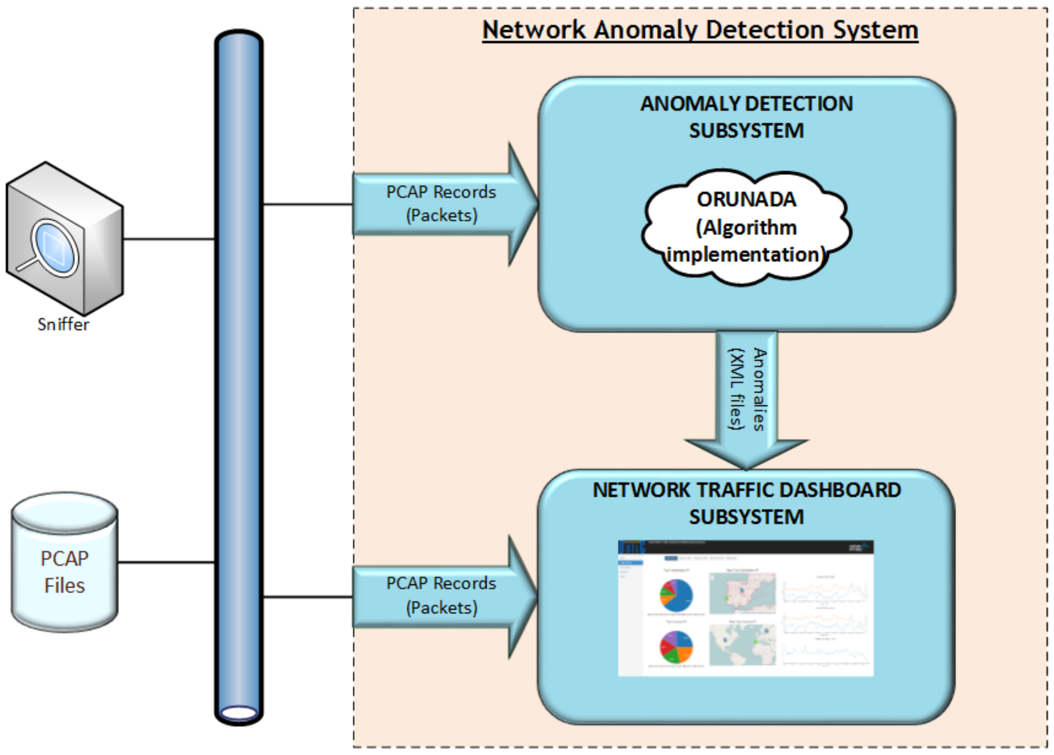

Network Anomaly Detection System
ANOMALY DETECTION SUBSYSTEM
ORUNADA (Algorithm implementation)
PCAP Records (Packets)
Sniffer
Anomalies (XML files)
NETWORK TRAFFIC DASHBOARD SUBSYSTEM
PCAP Files
PCAP Records (Packets)

Figure 1: Use Case #1 High-level Architecture

The Anomaly Detection subsystem analyses the network traffic that is captured at the PCAP format and applies the ORUNADA unsupervised anomaly detection algorithm presented in deliverable D4.2 [1]. When it detects an anomaly, it sends an XML file passing on the characteristics of the detected anomalies to the dashboard for display purposes. It can also be used for autonomously launching counter-measures.

The ORUNADA implementation has been carried out in Java (1.8 version).[1] ORUNADA detects anomalies based on traffic inputs in the PCAP files in a continuous way. It also generates signatures to describe them. ORUNADA outputs on the standard output some information about ORUNADA's execution. It also creates an XML file for each micro-slot processed (apart for the n first micro-slots, n equals to the number of micro-slots in a window). This XML structure lists the anomalous flows found in the PCAP file at the end of each micro-slot considering the packets contained in the current window. For each anomalous flow it specifies its features, its score of dissimilarity and its signature.

The Network Traffic Dashboard subsystem implements an ISP/CSP network administration tool that provides complete online (real-time) and offline (through a forensic analysis tool) traffic monitoring as well as anomaly detection.

The subsystem relies on both data traffic (PCAP and NetFlow V5) and the XML output of the Anomaly Detection subsystem. These input sources are processed through the Network Data Processing Module made up of four functional stages: ingestion, parsing, filtering and storage. On the other hand, the Visualization Module queries the stored data to display the analytics in Real-Time on a web user interface.

Such scenario requires the Network Data Processing Module to be supported by a series of characteristics related to performance and scalability in order to deal with the demands of a

[1] Code repository available in https://gitlab.com/ontic-wp4/ORUNADA

real deployment, i.e. to monitor a real network link. Some of these characteristics are: capability for processing high volumes of input data without losses, parallel processing of all input streams, distributed scalable and highly available system, near Real-Time support (in both Network Data Processing and Visualization modules). Hence, the Network Data Processing Module needs to be able to handle an increasing volume of work, which means it should improve its performance if the input load grows. Obviously, it will also reduce performance by contracting resource consumption, if necessary.

The point of convergence between the two modules of the dashboard subsystem is the database where data will be stored and queried. Due to the need of horizontal scaling to clusters of machines, which is a problem for relational databases, plus the heterogeneity of the data structures used, and a bunch of fast operations required for near Real-Time processing (like complex searches), a NoSQL database suits better in this case than a typical relational database.

Elasticsearch[2] is a highly scalable full-text search and analytics engine. It allows to store, search, and analyze big volumes of data quickly and in near Real-Time. Thus it fits as the underlying engine/technology to power the dashboard subsystem’s needs for storage and analysis.

7.1.3 Interworking and data model adaptation

Taking into account that the prototype environment does not provide a NetFlow input data source, the decision to integrate a module to generate NetFlow records from the PCAP file (in real-time) was made. This module works adding a NetFlow data stream as a new input data source to the dashboard subsystem. The output interface defined for the Anomaly Detection subsystem is a XML generator (it generates XML files periodically at specified time intervals). Each XML contains a list of attributes that define the anomalies detected in the period.

The dashboard subsystem receives and processes the XML files as soon as they arrive. Two such interface means are provided: through files written into a defined file directory or through a Web Service interface implemented in the dashboard system XMLs files could be sent continuously to.

The following DTD defines the legal building blocks of the XML's file sent by the LAAS-CNRS to SATEC. It describes the document structure with a list of legal elements and attributes. The DTD is associated with the XML document by means of a document type declaration (DOCTYPE):

[2] https://www.elastic.co/

```
<!DOCTYPE UNADA SYSTEM "/path/to/file.dtd">
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE UNADA[
<!ELEMENT UNADA (signaturesOfAnomalies*,points*)>
<!ELEMENT points (point+)>
<!ELEMENT point (val*)>
<!ELEMENT val (#PCDATA)>
<!ELEMENT signaturesOfAnomalies (signatureOfAnAnomaly*)>
<!ELEMENT signatureOfAnAnomaly (rule+)>
<!ELEMENT rule EMPTY >
<!ATTLIST UNADA start CDATA #REQUIRED>
<!ATTLIST UNADA end CDATA #REQUIRED>
<!ATTLIST UNADA file CDATA #REQUIRED>
<!ATTLIST UNADA aggreg CDATA #REQUIRED>
<!ATTLIST UNADA totalSize CDATA #REQUIRED>
<!ATTLIST UNADA totalNbPackets CDATA #REQUIRED>
<!ATTLIST point id CDATA #REQUIRED>
<!ATTLIST val dim CDATA #REQUIRED>
<!ATTLIST signatureOfAnAnomaly dissim CDATA #REQUIRED>
<!ATTLIST signatureOfAnAnomaly mainIPs CDATA #REQUIRED>
<!ATTLIST signatureOfAnAnomaly point CDATA #REQUIRED>
<!ATTLIST signatureOfAnAnomaly possAnom CDATA #REQUIRED>
<!ATTLIST rule dim CDATA #REQUIRED>
<!ATTLIST rule type CDATA #REQUIRED>
<!ATTLIST rule value CDATA #REQUIRED>
]>
```

Additionally, in order to provide the results of Use Case #1 to the network administrators a Graphical User Interface (GUI) has been developed as well. This GUI is a web-based application with the following features:

- It is a network traffic analytic dashboard application.
- It provides different views to show details about: traffic, flows and anomalies
- It works in both Online mode (real-time) and off-line mode (forensic mode)
- It enables select time periods and export data of that period.
- It can send message alerts to other systems in form of SNMP Traps.

7.1.4 Configuration and parametrization

ORUNADA takes 3 mandatory arguments plus 4 optional ones. The three mandatory arguments are:

- The path to the pcap file to analyze.
- The direction of the aggregation. You have to specify whether the aggregation is made at the IP source 'src' or at the IP destination 'dst'.
- The mask of the aggregation. You need to specify if it is '8', '16', '24', '32'.

The four optional arguments are:

- Time of a slot in seconds;
- Nb of micro-slots in a slot;

- For computing the size of the intervals for IGDCA. Each dimension has a different size of interval. This value is set as a percentage of the maximum distance between every pair of points for each dimension. To fix them, you need to specify this percentage and specify a value between 1 and 99.");
- For computing the minimum number of points to form a cluster in IGDCA. This number is set as a percentage of the whole number of points. To fix it, you need to specify this percentage and specify a value between 1 and 99."); If you don't provide the four optional arguments, some defaults ones are used.

The dashboard is a web application that does not need any special parametrization.

7.1.5 Open issues and future developments

The ORUNADA algorithm has been implemented and assessed. It exhibits real-time performance when applied to ONTS traffic. It also exhibits high detection accuracy on all ground truths that we used for testing it. ORUNADA then fits the ONTIC objectives. Future work will be related to the relations established by ONTIC with the H2020 EBDEAVOUR project, Indeed, IBM Zürich, one of the ENDEAVOUR partners, is interested by ORUNADA for its own purpose. The same evaluation with IBM traffic has to be run as for the ONTS traffic in ONTIC.

The open lines identified for dashboard further progress include:

- Analysing new input data sources supplying additional information about anomalies such as SNMP traps and logs from network appliances, anomaly signatures from data bases, etc.
- Prospecting, selecting and integrating of a real-time network traffic capturing system to be integrated as part of the product.
- Development of interfaces for integrating real-time network traffic capturing systems such as the ONTS Provisioning System developed by the project or other commercially available systems.
- Designing and implementing new analytics processes taken into account the new data sources.

7.2 Use case #2: Adaptive Quality of Experience Control

7.2.1 Use case description

Use Case #2 deals with the detection on QoE degradation situations and the subsequent actuation to alleviate the degradation situation. A specific type of service, Video, has been selected for this proof-of-concept, and a test-bed has been devised to assess the feasibility of the overall schema. As the ONTS dataset does not carry application-level information, a smaller, labeled dataset has been generated by the partners so that it can be used to train the clustering model the algorithms rely upon. New network traces generated by users' traffic are matched against the model and assigned to a cluster. Depending on the weights of the labels in each cluster, traces are given a numeric value, which helps to assess the degradation of QoE.

7.2.2 Implementation details

As outlined in the previous section, Use Case #2 aims to detect degradations in the QoE perceived in video services. The implementation of this use case relies on an adaptation of FreeScan. The algorithm as such has been implemented in Python and customized to the requirements of the use case, as domain-specific constraints apply (for instance, the location segmentation or the use of time windows to continuously monitor the QoE indicators and to report end-of-degradation events).

The overall procedure is outlined in the figure below:

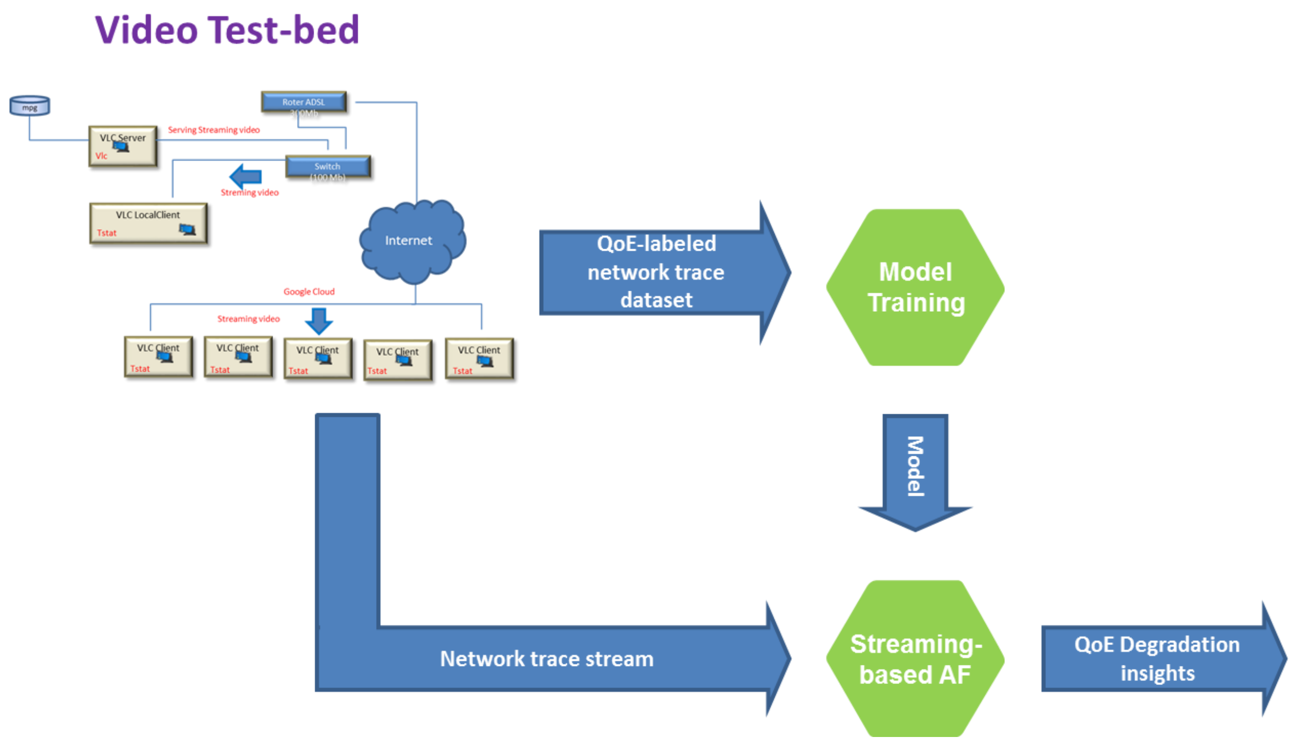


Figure 2: Use Case #2 Overall Schema

Use Case #2 relies on capturing network traces and using them to detect QoE degradation situations. This task is carried out by the Network Trace Forwarder and is based on Tstat [1], which aggregates the actual network traces into flows. The output of Tstat is a set of CVS files where each row corresponds to a different flow and each column is associated to a specific observed measure in the flow. In order to train the model, DellEMC has manually labeled a network trace dataset, where each flow is given a ‘good’, ‘medium’, ‘bad’ label. Once the clusters have been found, clusters are examined in order to determine whether they have a straightforward relationship to the QoE labels. As QoE labels are categorical variables, each of them is assigned a numeric value and, depending on the label share in each cluster, a QoE value is assigned to each cluster.

The numeric values are defined according to the eleven-grade numeric quality scale proposed by ITU-T [11]. ‘Excellent’ (9) has been mapped to “Good”, ‘Fair’ (5) to “Medium”, and ‘Poor’ (3) to “Bad”. People on charge of labelling was illustrated with the meaning of the label during the training phase.

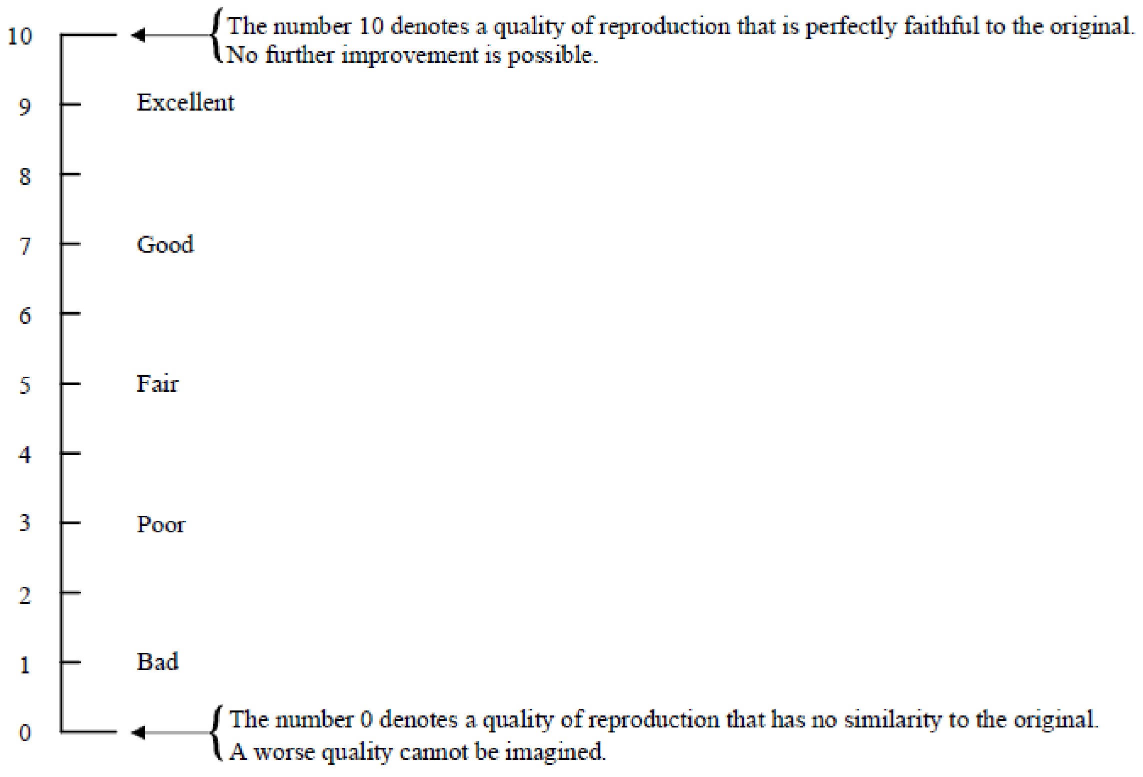

10
The number 10 denotes a quality of reproduction that is perfectly faithful to the original. No further improvement is possible.
9
Excellent
8
7
Good
6
5
Fair
4
3
Poor
2
1
Bad
0
The number 0 denotes a quality of reproduction that has no similarity to the original. A worse quality cannot be imagined.

Figure 3: Eleven-grade numeric quality scale proposed by ITU-T (*Subjective video quality assessment methods for multimedia applications*)

When the model is trained, it is applied to each new network trace and a QoE numeric value is therefore assigned to the trace. When the average value of this QoE indicator for a given time window goes below a threshold for more than a predefined time, an alarm is issued and a mitigation plan is applied so that the degradation situation is alleviated. The computation of this indicator, `video_qoe`, is done in 30-second windows. Computation is repeated every 15 seconds, considering the last 30-second window.

A detailed description of network traces, model build and QoE degradation detection can be found in Annex A.

7.2.3 Interworking and data model adaptation

The centralized Analytics Function that implements the ONTIC algorithms plays the role of message broker consumer and receives the networks traces as plain text according to the Tstat format. The outcome of the Analytics Function implements the IF3-2bis, with payload described in Annex C.2 in deliverable D5.6 [5].

7.2.4 Configuration and parametrization

For simplicity and according to the teams' capabilities, Python has been the selected programming language to implement the ONTIC algorithms. The Analytics Function implementation is based on Spark 1.6.0 using Python (PySpark) and takes advance of Spark Streaming, an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. The clustering model the algorithm relies upon is trained by means of the `pyspark.mllib.clustering` module from the `pyspark.mllib` package.[3] Once the model has been trained, it is loaded by the Spark Streaming logic (see below). The model is used only to classify incoming traffic, but not to determine when QoE trespasses a threshold. Auxiliary logic to so has been developed.

Provided that the model is loaded and is made available to logic deployed in Spark Streaming, a Spark Context is created and a Streaming Context is associated to it. The Streaming Context is connected to the Kafka message broker and it is subscribed to the network traces available in the 'analytics-input' topic.

The parameters in the configuration file are the following ones:

```
{
    "kafkaLocation":"kafka:9092",
    "kafkaLocationForPGF":"kafka:9092",
    "dataTopic":"analytics-input",
    "pgfTopic":"analytics-to-rest",
    "checkpointKafka":"checkpoint",
    "streamBatchSize":9,
    "windowSize":36,
    "execTime": 18,
    "model_path": "model"
}
```

Spark Streaming follows a micro-batch schema. The size of the batch is configured by means of `streamBatchSize`, in seconds. `windowSize` is the window length, the duration of the window when performing windowed computations. `execTime` is the sliding interval, the interval at which the window operation is performed. Both parameters must be multiples of the batch interval of the source stream (`streamBatchSize`).

The code below summarizes the adaptation of the ONTIC algorithms to be executed on Spark Streaming. First we load the configuration parameters, create the Spark StreamingContext (the main entry point for all streaming functionality), associate it to Kafka to receive data and get a `DStream` object (discretized stream, a high-level Spark abstraction representing a continuous stream of data), and load the trained model:

```
# Configuration load
with open('project.conf') as data_file:
    config = json.load(data_file)

# Create SparkContext() and StreamingContext(), with batch size from
configStreaming.
sc = SparkContext()
ssc = StreamingContext(sc, config["streamBatchSize"])

# Kafka topic where streaming is reading and checkpoint definition.
kafkaParams = {"metadata.broker.list": config["kafkaLocation"]}

# Connect to Kafka in order to get the network traces from the Network Trace
Forwarding
while True:
    try:
```

[3] https://spark.apache.org/docs/1.6.0/api/python/pyspark.mllib.html

```
        directKafkaStream = KafkaUtils.createDirectStream(ssc,
                                                           [config["dataTopic"]],
                                                           kafkaParams)
        break
    except Exception as e:
        print "Error: Waiting for kafka...".format(e)
        time.sleep(5)
```

```
ssc.checkpoint(config["checkpointKafka"])
qoe_model = KMeansModel.load(sc, config["model_path"])
```

Next, the sequence of operations to be performed to the network traces is defined. Transformation functions are defined in a separated file (**`dataformatfunctions.py`**), which is imported as the `dft` module. These transformations are applied to the DStream object in three different steps (variable selection within the network trace, string to float transformation, and indicator generation from the features in network traces).

```
init = directKafkaStream.map(lambda x:x[1])
init = init.map(dft.str_to_list)
KPIs = init.map(dft.get_dimensions)/
          .map(dft.to_float)/
          .map(dft.kpi_generation)
```

To get the group shares, the information is extracted from the `DStream` object and it is summarized on a Python dictionary to send it later to the PGF-Adaptor.

```
groups = init.map(lambda x : x[0])
groups = groups.map(lambda group:(group,1))\
    .reduceByKeyAndWindow(lambda a,b : a+b,
                          lambda a,b : a-b,
                          config["windowSize"],
                          config["execTime"]
                          )\
    .reduce(dft.to_array)\
    .map(dft.get_percentages)
```

With regard to the computation of the `qoe_video` indicator, the clustering model is applied to the KPIs obtained in previous steps. Thus, we obtain the cluster the traces are assigned to calculate the average value in the micro-batch.

```
# Data is tagged depending on the cluster it belongs
result1 = KPIS.map(qoe_model.predict)
```

```
# KPI is calculated depending on how many sessions are tagged as good
result = result1.map(lambda tag: (tag, 1))\
                .reduceByKeyAndWindow(lambda x,y : (x+y),
                                      lambda x,y : x-y,
                                      config["windowSize"],
                                      config["execTime"]
                                      )\
                .reduce(dft.toArray)
result = result.map(dft.get_summary) \
    .union(groups) \
    .reduce(dft.to_json)
```

The last step is passing on the information to the PGF Adaptor by means of the Kafka message broker. It is the PGF Adaptor the entity that decides if the data obtained is sensitive enough to raise an alert.

```
result.foreachRDD (lambda x :sendkafka(x,
                                      config["kafkaLocationForPGF"],
                                      config["pgfTopic"])
                   )
```

Note that the steps above only instruct Spark Streaming to set up the computation it will perform when it is started, and no real processing has started yet. To start the processing, the Spark Streaming Context has to be actually started and wait for its termination.

```
ssc.start()
ssc.awaitTermination()
```

7.2.5 Open issues and future developments

Future developments will address two different areas: first of all, the application of the same techniques to other types of (video) services. The main challenge involved here lies in the need to manually label large amounts of network traffic traces to be able to subsequently detect QoE degradations. It is assumed that a degradation in a specific set of services has the same root cause and therefore no labeling for every type of services is needed. On the other hand, the adaptation and configuration of the ONTIC algorithms to include domain-specific constraints and to provide further flexibility.

7.3 Use case #3: Proactive Congestion Detection and Control

7.3.1 Use case description

Nowadays, in the Internet, end to end host mechanisms for congestion control such as TCP are widely deployed, scale to existing traffic loads, and share network bandwidth applying a flow-based fairness. However, in a context where it is expected, in the short term, that more than 90 percent of the Internet traffic will go through data centers, TCP converge slowly to steady sending rates and cannot isolate datacenter tenants from interfering with each other.

In this scenario, UC #3 proposes a combined deployment of proactive congestion control protocols explicitly computing sending rates and rate enforcement points placed in the edges of the network as a solution to establish the right way to mitigate these issues. This use case aims to meet two complementary requirements: First, to detect in advance congestion problems that could occur in the network links by means of advanced forecasting mechanisms, and second, to avoid these congestion problems predicting max-min fair sending rates and assigning them to network sessions.

The functionalities developed in this use case contains the following elements: (a) a scalable and distributed max-min fair optimization algorithm (MMFA) to compute max-min fair rates that iterates rapidly until converging to the optimal solution, (b) a proactive end-to-end congestion control protocol (EERCP) that applies the former max-min algorithm to compute the session sending rates and enforces the fulfillment of these rates at the edges of the network and (c) a machine learning component that implements forecasting capabilities.

EERCP and MMFA components are an extension of the SLBN protocol, a proactive EERC protocol that is also scalable. We extended SLBN to be able to transport and manage rate max-min fair rate predictions. Regarding the forecasting module, in a first phase we trained and tested a linear regression model and later a more complex and powerful deep neural network architecture based on Convolutional Neural Networks was integrated. Moreover, the design of this forecasting component is agnostic of the specific technique to be used (e.g. time series, ARIMA, linear regression, neural networks) and so, depending on the problem complexity, different models can be trained and plugged in. A detailed description of this use case can be found in deliverable D5.6. [6]

We demonstrate the proposed solution by means of simulations since nowadays it is unfeasible, at least in a first phase, to setup a realistic deployment of new congestion control solutions involving hundreds of routers and thousands of nodes connected to a network.

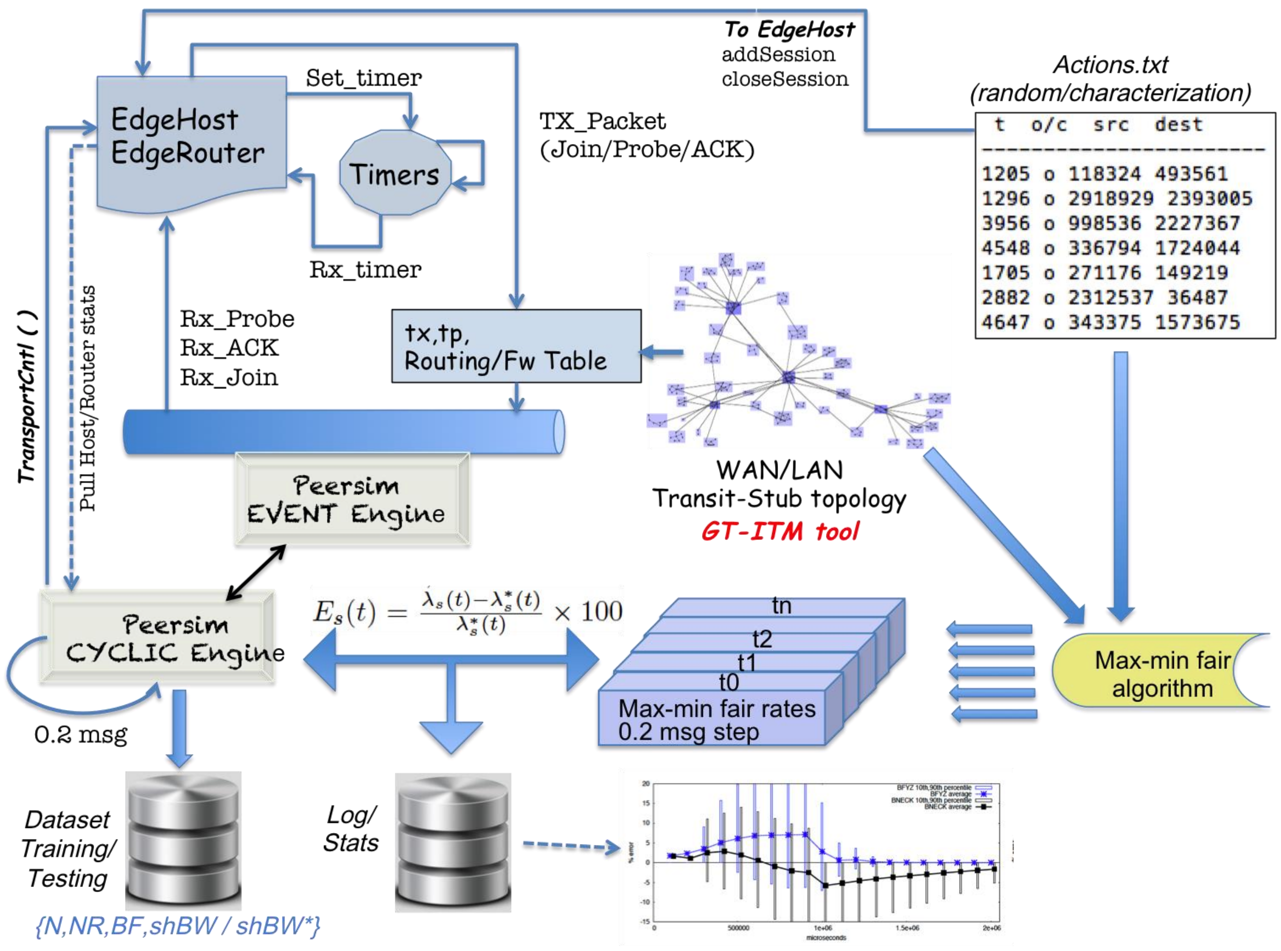
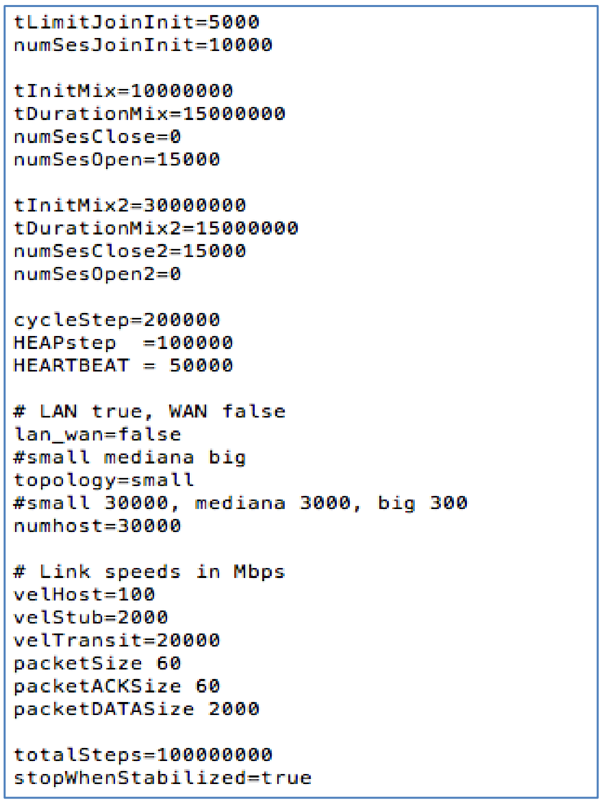


Figure 4: Global perspective of the Use Case #3 Simulator.

7.3.2 Implementation details

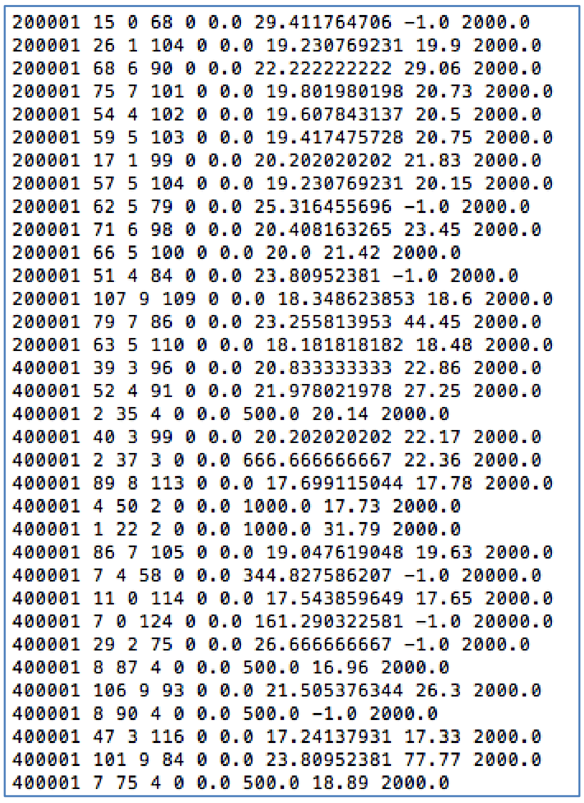
Regarding the unfeasibility of a realistic deployment for congestion control solutions, UC#3 will be demonstrated by means of simulations run on top of an extended version of Peersim,[12] a P2P discrete event simulator. We thoroughly modified and extended Peersim in order to (a) run experiments consisting of thousands of routers and up to a million of hosts and sessions, (b) import Internet-like topologies generated with the Georgia Tech gt-itm tool;[13] and (c) model different network elements not present in Peersim such us transmission and propagation times in the network links (e.g., LAN or WAN configurations), processing time in routers and limited size packet in link queues. Figure 4 shows a global perspective of the set of extensions we added to Peersim in order to setup the required scenario to demonstrate this use case. Specifically, the EERCP, MMFA and FM modules are implemented in EdgeRouter and EdgeHost components. A detailed description of the use case implementation can be found in deliverable D5.6. [6]

The proposed simulator accepts as input a configuration file (Figure 5) in which we define the parameters that describe the simulation to be run. In addition, the simulator generates as output dataset files (Figure 6) for retraining and testing the forecasting model and text files containing logs and statistics (Figure 7). The latter can be processed with graphical tools in order to plot the protocol behavior. We show in Figure 8 some plots of the distribution of the queue size of router links and in Figure 9 the accuracy in the rate assignments, both of which were obtained from an output log file.



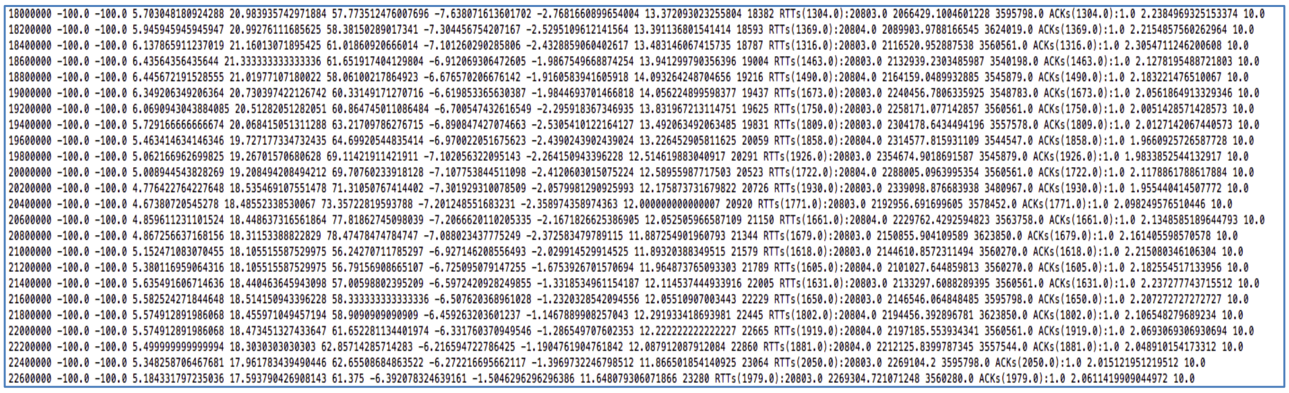
```
tLimitJoinInit=5000
numSesJoinInit=10000

tInitMix=10000000
tDurationMix=15000000
numSesClose=0
numSesOpen=15000

tInitMix2=30000000
tDurationMix2=15000000
numSesClose2=15000
numSesOpen2=0

cycleStep=200000
HEAPstep  =100000
HEARTBEAT = 50000

# LAN true, WAN false
lan_wan=false
#small mediana big
topology=small
#small 30000, mediana 3000, big 300
numhost=30000

# Link speeds in Mbps
velHost=100
velStub=2000
velTransit=20000
packetSize 60
packetACKSize 60
packetDATASize 2000

totalSteps=100000000
stopWhenStabilized=true
```

Figure 5: Example of a configuration file



```
200001 15 0 68 0 0.0 29.411764706 -1.0 2000.0
200001 26 1 104 0 0.0 19.230769231 19.9 2000.0
200001 68 6 90 0 0.0 22.222222222 29.06 2000.0
200001 75 7 101 0 0.0 19.801980198 20.73 2000.0
200001 54 4 102 0 0.0 19.607843137 20.5 2000.0
200001 59 5 103 0 0.0 19.417475728 20.75 2000.0
200001 17 1 99 0 0.0 20.202020202 21.83 2000.0
200001 57 5 104 0 0.0 19.230769231 20.15 2000.0
200001 62 5 79 0 0.0 25.316455696 -1.0 2000.0
200001 71 6 98 0 0.0 20.408163265 23.45 2000.0
200001 66 5 100 0 0.0 20.0 21.42 2000.0
200001 51 4 84 0 0.0 23.80952381 -1.0 2000.0
200001 107 9 109 0 0.0 18.348623853 18.6 2000.0
200001 79 7 86 0 0.0 23.255813953 44.45 2000.0
200001 63 5 110 0 0.0 18.181818182 18.48 2000.0
400001 39 3 96 0 0.0 20.833333333 22.86 2000.0
400001 52 4 91 0 0.0 21.978021978 27.25 2000.0
400001 2 35 4 0 0.0 500.0 20.14 2000.0
400001 40 3 99 0 0.0 20.202020202 22.17 2000.0
400001 2 37 3 0 0.0 666.666666667 22.36 2000.0
400001 89 8 113 0 0.0 17.699115044 17.78 2000.0
400001 4 50 2 0 0.0 1000.0 17.73 2000.0
400001 1 22 2 0 0.0 1000.0 31.79 2000.0
400001 86 7 105 0 0.0 19.047619048 19.63 2000.0
400001 7 4 58 0 0.0 344.827586207 -1.0 20000.0
400001 11 0 114 0 0.0 17.543859649 17.65 2000.0
400001 7 0 124 0 0.0 161.290322581 -1.0 20000.0
400001 29 2 75 0 0.0 26.666666667 -1.0 2000.0
400001 8 87 4 0 0.0 500.0 16.96 2000.0
400001 106 9 93 0 0.0 21.505376344 26.3 2000.0
400001 8 90 4 0 0.0 500.0 -1.0 2000.0
400001 47 3 116 0 0.0 17.24137931 17.33 2000.0
400001 101 9 84 0 0.0 23.80952381 77.77 2000.0
400001 7 75 4 0 0.0 500.0 18.89 2000.0
```

Figure 6: Example of a dataset file.
Lines are generated at periodic intervals. Fields are as follow: time_stamp, link identification (node_src, node_dst), N, R, BF, shBW and C



```
18000000 -100.0 -100.0 5.703048180924288 20.983935742971884 57.773512476007696 -7.638071613601702 -2.7681660899654004 13.372093023255804 18382 RTTs(1304.0):20803.0 2066429.1004601228 3595798.0 ACKs(1304.0):1.0 2.2384969325153374 10.0
18200000 -100.0 -100.0 5.945945945945947 20.99276111685625 58.38150289017341 -7.304456754207167 -2.5295109612141564 13.391136801541414 18593 RTTs(1369.0):20804.0 2089903.9788166545 3624019.0 ACKs(1369.0):1.0 2.215485756026296 4 10.0
18400000 -100.0 -100.0 6.137865911237019 21.16013071895425 61.01860920666014 -7.101260290285806 -2.4328859060402617 13.483146067415735 18787 RTTs(1316.0):20803.0 2116520.952887538 3560561.0 ACKs(1316.0):1.0 2.305471124620060 8 10.0
18600000 -100.0 -100.0 6.43564356435644 21.333333333333336 61.651917404129804 -6.912069306472605 -1.9867549668874254 13.941299790356396 19004 RTTs(1463.0):20803.0 2132939.2303485987 3540198.0 ACKs(1463.0):1.0 2.1278195488721803 10.0
18800000 -100.0 -100.0 6.445672191528555 21.01977107180022 58.06100217864923 -6.676570206676142 -1.9160583941605918 14.093264248704656 19216 RTTs(1490.0):20804.0 2164159.0489932885 3545879.0 ACKs(1490.0):1.0 2.18322147651006 7 10.0
19000000 -100.0 -100.0 6.349206349206364 20.730397422126742 60.33149171270716 -6.619853365630387 -1.9844693701466818 14.056224899598377 19437 RTTs(1673.0):20803.0 2240456.7806335925 3548783.0 ACKs(1673.0):1.0 2.0561864913329346 10.0
19200000 -100.0 -100.0 6.0690943043884085 20.51282051282051 60.864745011086484 -6.700547432616549 -2.295918367346935 13.831967213114751 19625 RTTs(1750.0):20803.0 2258171.077142857 3560561.0 ACKs(1750.0):1.0 2.005142857142857 3 10.0
19400000 -100.0 -100.0 5.72916666666674 20.068415051311288 63.21709786276715 -6.890847427074663 -2.5305410122164127 13.492063492063485 19831 RTTs(1809.0):20803.0 2304178.6434494196 3557578.0 ACKs(1809.0):1.0 2.0127142067440573 10.0
19600000 -100.0 -100.0 5.463414634146346 19.727177334732435 64.69920544835414 -6.970022051675623 -2.4390243902439024 13.226452905811625 20059 RTTs(1858.0):20804.0 2314577.815931109 3544547.0 ACKs(1858.0):1.0 1.9660925726587728 10.0
19800000 -100.0 -100.0 5.062166962699825 19.26701570680628 69.1142191142191 1 -7.102056322095143 -2.264150943396228 12.514619883040917 20291 RTTs(1926.0):20803.0 2354674.9018691587 3545879.0 ACKs(1926.0):1.0 1.9833852544132917 10.0
20000000 -100.0 -100.0 5.008944543828269 19.208494208494212 69.70760233918128 -7.107753844511098 -2.4120603015075224 12.58955987717503 20523 RTTs(1722.0):20804.0 2288005.0963995354 3560561.0 ACKs(1722.0):1.0 2.1178861788617884 10.0
20200000 -100.0 -100.0 4.776422764227648 18.535469107551478 71.31050767414402 -7.301929310078509 -2.0579981290925993 12.175873731679822 20726 RTTs(1930.0):20803.0 2339098.876683938 3480967.0 ACKs(1930.0):1.0 1.95544041450777 2 10.0
20400000 -100.0 -100.0 4.67380720545278 18.48552338530067 73.3572281959378 8 -7.20124855168323 1 -2.358974358974363 12.000000000000007 20920 RTTs(1771.0):20803.0 2192956.691699605 3578452.0 ACKs(1771.0):1.0 2.09824957651044 6 10.0
20600000 -100.0 -100.0 4.859611231101524 18.448637316561864 77.81862745098039 -7.206662011020533 5 -2.1671826625386905 12.052505966587109 21150 RTTs(1661.0):20804.0 2229762.4292594823 3563758.0 ACKs(1661.0):1.0 2.1348585189644793 10.0
20800000 -100.0 -100.0 4.867256637168156 18.31153388822829 78.47478474784747 -7.088023437775249 -2.372583479789115 11.887254901960793 21344 RTTs(1679.0):20803.0 2150855.904109589 3623850.0 ACKs(1679.0):1.0 2.161405598570578 10.0
21000000 -100.0 -100.0 5.152471083070455 18.105515587529975 56.24270711785297 -6.927146208556493 -2.029914529914525 11.89320388349515 21579 RTTs(1618.0):20803.0 2144610.8572311494 3560270.0 ACKs(1618.0):1.0 2.215080346106304 10.0
21200000 -100.0 -100.0 5.380116959064316 18.105515587529975 56.7915690866510 7 -6.725095079147255 -1.6753926701570694 11.964873765093303 21789 RTTs(1605.0):20804.0 2101027.644859813 3560270.0 ACKs(1605.0):1.0 2.182554517133956 10.0
21400000 -100.0 -100.0 5.635491606714636 18.440463645943098 57.00598802395209 -6.5972420928249855 -1.3318534961154187 12.114537444933916 22005 RTTs(1631.0):20803.0 2133297.6088289395 3560561.0 ACKs(1631.0):1.0 2.237277743715512 10.0
21600000 -100.0 -100.0 5.582524271844648 18.514150943396228 58.333333333333336 -6.507620368961028 -1.2320328542094556 12.0551090700 3443 22229 RTTs(1650.0):20803.0 2146546.064848485 3595798.0 ACKs(1650.0):1.0 2.207272727272727 10.0
21800000 -100.0 -100.0 5.574912891986068 18.455971049457194 58.909090909090 9 -6.459263203601237 -1.1467889908257043 12.291933418693981 22445 RTTs(1802.0):20804.0 2194456.392896781 3623850.0 ACKs(1802.0):1.0 2.106548279689234 10.0
22000000 -100.0 -100.0 5.574912891986068 18.473451327433647 61.652281134401974 -6.331760370949546 -1.2865497076023 53 12.222222222222227 22665 RTTs(1919.0):20804.0 2197185.553934341 3560561.0 ACKs(1919.0):1.0 2.0693069306930694 10.0
22200000 -100.0 -100.0 5.499999999999994 18.303030303030303 62.85714285714283 -6.216594722786425 -1.1904761904761842 12.087912087912084 22860 RTTs(1881.0):20804.0 2212125.8399787345 3557544.0 ACKs(1881.0):1.0 2.04891015417331 2 10.0
22400000 -100.0 -100.0 5.348258706467681 17.961783439490446 62.65508684863522 -6.272216695662117 -1.3969732246798512 11.866501854140925 23064 RTTs(2050.0):20803.0 2269104.2 3595798.0 ACKs(2050.0):1.0 2.015121951219512 10.0
22600000 -100.0 -100.0 5.184331797235036 17.593790426908143 61.375 -6.392078324639161 -1.5046296296296386 11.648079306071866 23280 RTTs(1979.0):20803.0 2269304.721071248 3560280.0 ACKs(1979.0):1.0 2.0611419909044972 10.0
```

Figure 7: Example of an output log file. Each line is generated at periodic intervals. Several percentiles of the session rate assignment errors are show. In addition, other statistical values (e.g., RTTs and ACK packets) are included in each line.

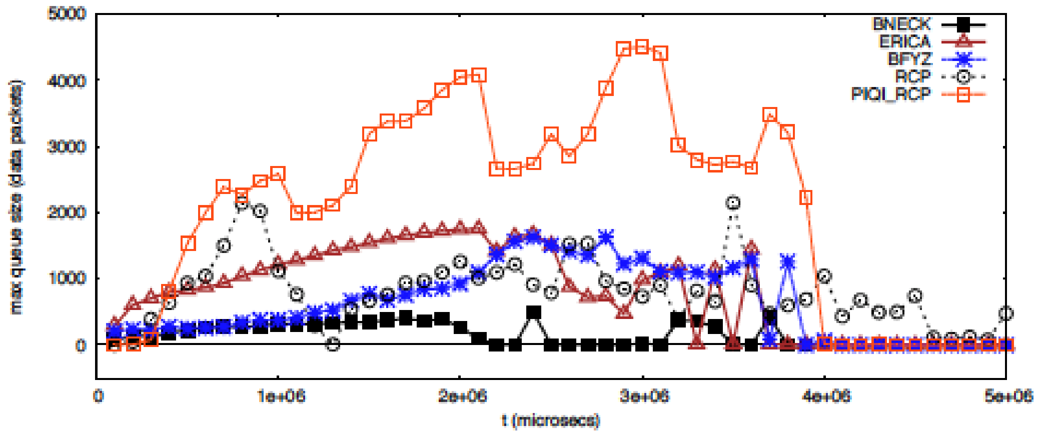


Figure 8: Plot examples of the queue size distribution of links in an experiment running several EERC protocols

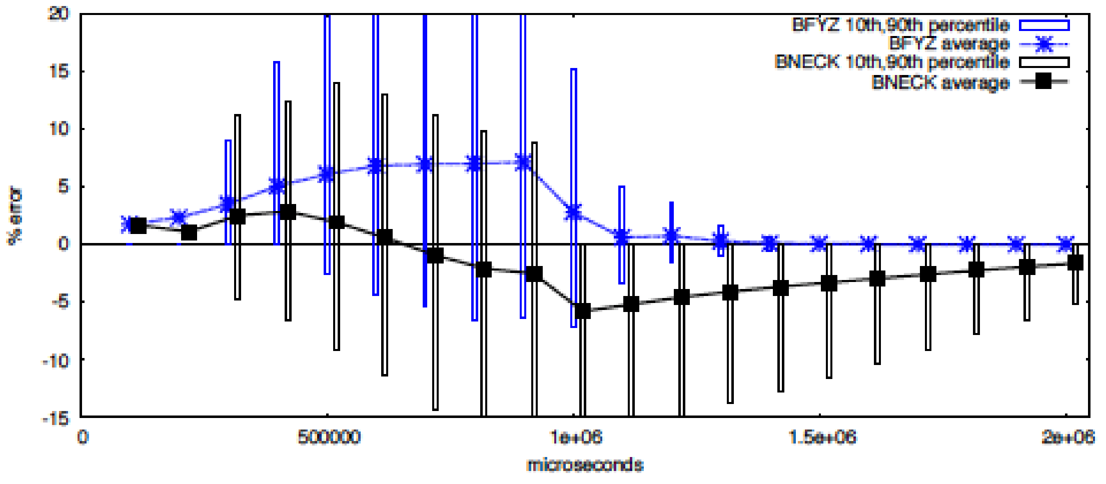
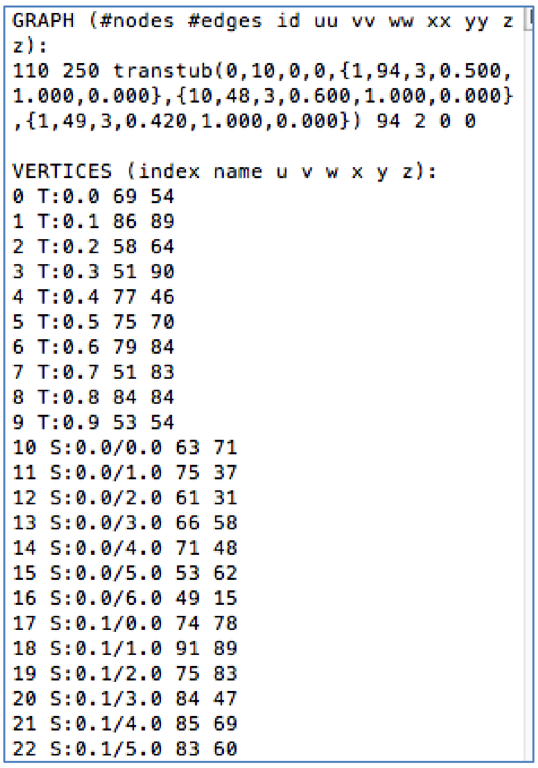


Figure 9. Plot example of the error distribution of rate assignments at sources in an experiment running BFYZ and BNECK EERC protocols

7.3.3 Interworking and data model adaptation

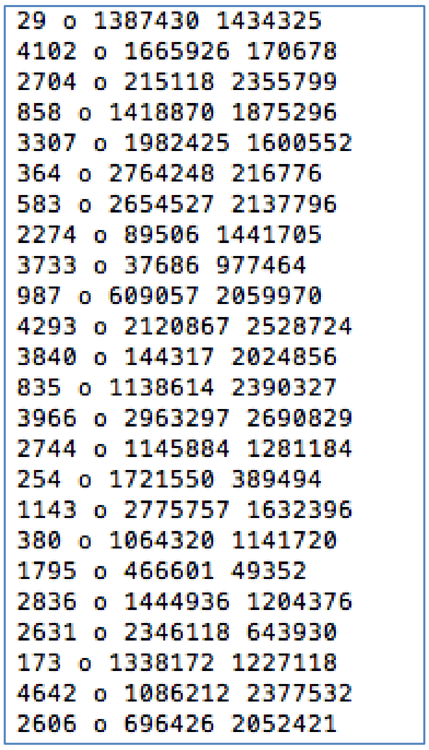
7.3.3.1 Simulator

From a global perspective the simulator is a standalone application coded in Java and so, it can be executed on top of a Java Virtual Machine. Therefore, the system has no interconnection with other systems except with the gt-itm tool for importing network topologies and the traffic pattern generator for importing the actions.txt file. Both tools generate text files in order to ease the import procedure. Figure 10 shows an example of the file "small.alt", generated with the gt-itm tool, and describing a Small topology. Figure 11 shows an example of the actions.txt file.



```
GRAPH (#nodes #edges id uu vv ww xx yy z
z):
110 250 transtub(0,10,0,0,{1,94,3,0.500,
1.000,0.000},{10,48,3,0.600,1.000,0.000}
,{1,49,3,0.420,1.000,0.000}) 94 2 0 0

VERTICES (index name u v w x y z):
0 T:0.0 69 54
1 T:0.1 86 89
2 T:0.2 58 64
3 T:0.3 51 90
4 T:0.4 77 46
5 T:0.5 75 70
6 T:0.6 79 84
7 T:0.7 51 83
8 T:0.8 84 84
9 T:0.9 53 54
10 S:0.0/0.0 63 71
11 S:0.0/1.0 75 37
12 S:0.0/2.0 61 31
13 S:0.0/3.0 66 58
14 S:0.0/4.0 71 48
15 S:0.0/5.0 53 62
16 S:0.0/6.0 49 15
17 S:0.1/0.0 74 78
18 S:0.1/1.0 91 89
19 S:0.1/2.0 75 83
20 S:0.1/3.0 84 47
21 S:0.1/4.0 85 69
22 S:0.1/5.0 83 60
```

Figure 10: Example of a "small.alt" file generated for a Small topology using the gt-itm tool. The topology is composed of transit (T) and stub (S) nodes.



```
29 o 1387430 1434325
4102 o 1665926 170678
2704 o 215118 2355799
858 o 1418870 1875296
3307 o 1982425 1600552
364 o 2764248 216776
583 o 2654527 2137796
2274 o 89506 1441705
3733 o 37686 977464
987 o 609057 2059970
4293 o 2120867 2528724
3840 o 144317 2024856
835 o 1138614 2390327
3966 o 2963297 2690829
2744 o 1145884 1281184
254 o 1721550 389494
1143 o 2775757 1632396
380 o 1064320 1141720
1795 o 466601 49352
2836 o 1444936 1204376
2631 o 2346118 643930
173 o 1338172 1227118
4642 o 1086212 2377532
2606 o 696426 2052421
```

Figure 11: Example of an "actions.txt" file. Fields are as follow: timestamp, o/c (open or close) and session id (src_node, dst_node)

7.3.3.2 PCAD (EERCP + MMFA) and FM modules

In a real deployment, the following components are proposed: a proactive EERC protocol (EERCP), a distributed max-min fair algorithm (MMFA) and a forecasting module (FM). Said functional components are deployed in network routers and in particular they are instantiated in each a router link. Additionally, in a realistic environment, enforcement points (EF) will be deployed in the edges of the network (e.g., in the edge routers that are connected with hosts) in order to monitor and control that all hosts observe the assigned bandwidth rates. Hosts run a simplified version of the EERCP.

From an architectural perspective (Figure 12), EERCP and MMFA are integrated in a single component named Proactive Congestion Avoidance Driver (PCAD) that integrates both functionalities. Each router link runs an instance of this driver jointly with the Forecasting Module. These three components interwork with the switching, forwarding and link layers present in a typical router. When an EERC protocol packet is received in a link, it is delivered to the PCAD module. After processing the packet, the PCAD send it to the routing plane to decide the output link where it should be forwarded.

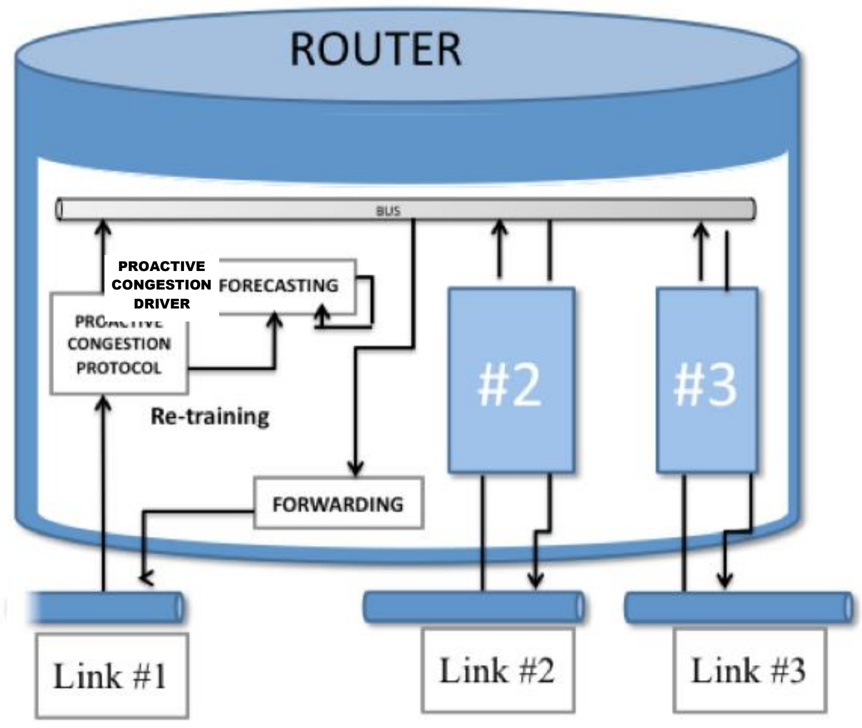


Figure 12: UC #3 System Architecture

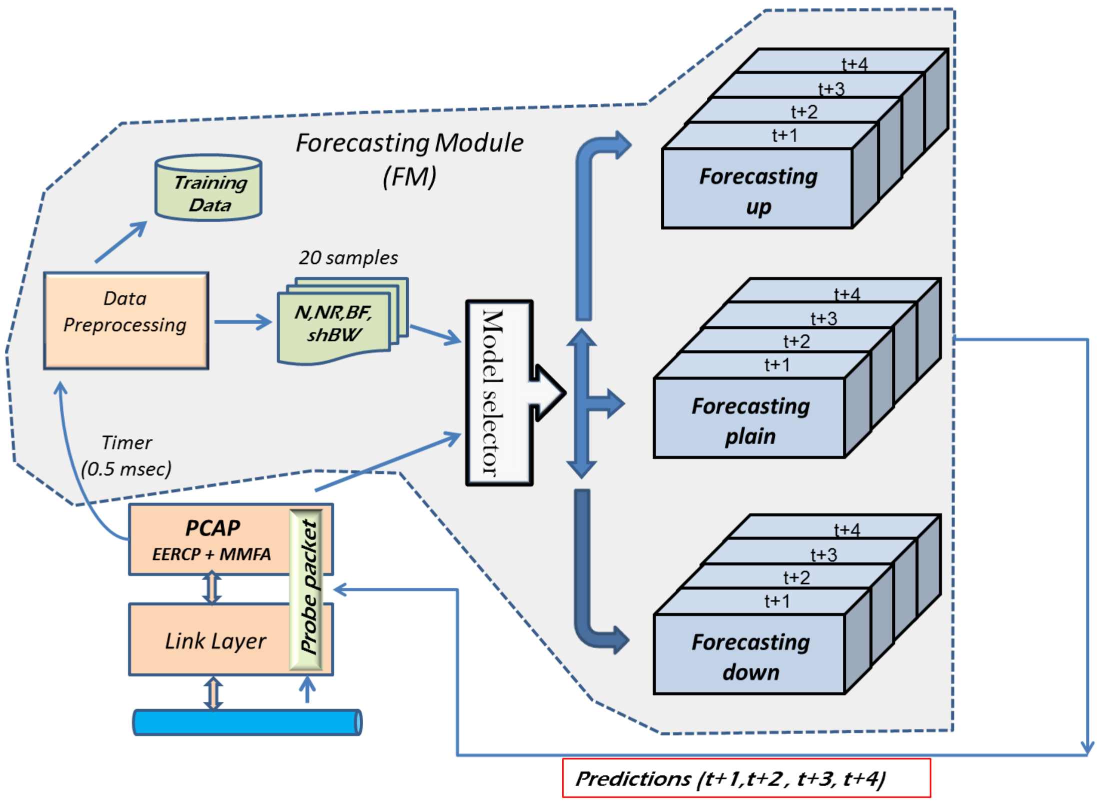


Figure 13: Detail of the integration of PCAP and FM modules.

The Forecasting Module (FM) is comprised of two modules: A Data Preprocessing (DP) unit and a Forecasting (FC) unit that are connected sequentially. The DP unit collects periodically N, NR, BF and shBW values from EERCP. These values are stored in a log file to be used later for subsequent training and testing of the model. These values are also stored to be utilized as input to the forecasting algorithms. These algorithms will compute future predictions of the max-min fair rates each time a Probe packet arrives at a router link. Figure 13 details the integration of PCAP (EERCP+MMFA) with the FM module. When a protocol packet arrives at a network link it is delivered to the PCAP module to be processed. After this processing is done and before forwarding the packet to the output link, PCAP activates the FM module to generate the corresponding predictions of the bottleneck value of this link. The Model Selector units decides which Forecasting unit should be activated (up, down or plain). After that, the activated unit generates the bottleneck predictions that are stored in the packet. Finally, the packet is forwarded to the output link.

7.3.4 Configuration and parametrization

The simulator accepts as input a configuration file in which we define the parameters that describe the experiment to be run. Figure 5 shows an example of this file. Three different time intervals can be defined:

(1) from 0 to tLimitJoinInit;

(2) from tInitMix with a duration of tDurationMix; and

(3) from tInitMix2 and with a duration of tDurationMix2.

At each interval different number of sessions joining and leaving can be configured (numSesJoinInit, numSesClose, NumSesOpen, numSesClose2 and NumSesOpen2). With these parameters the simulator creates an “actions.txt” file (Figure 11) to be used during the simulation. The network topology (topology=Small, Medium or Big), WAN/LAN configuration (lan_wan= true, false) link speeds (velHost, velStub and velTransit) and packet sizes (are also configured in this file. In addition, we can define the log intervals (cyclesStep) and finalization conditions. For example, a time limit (totalSteps) or the protocol convergence to max-min fair rates (stopWhenStabilized = true, false).

The PCAD does not need to be configured with any parameter to run. Finally, the initialization of the Forecasting module (FM) is done within the class constructor method for the corresponding algorithm. In the case of the Linear Regression module, the initialization procedure reads the linear regression coefficients for each forecasting model (up, down and plain) and network link (e.g., the files u-p-103-9, d-p-103-9, p-p-103-9 for link 103-9). Each file contains the linear regression coefficients (80 values = 20 samples* 4 variables -N, NR, BF and shBW-) for each predictor (e.g. 4 predictors, t+1, t+2, t+3 and t+4 in the example of the Figure 13).

7.3.5 Open issues and future developments

Next steps will consider a more realistic deployment in a network laboratory with real routers and hosts. In addition, we plan to train and test more complex forecasting algorithms. We think that Convolutional neural networks with a multilayer architecture can take advantage of the temporal nature of the data available in this use case. Since we have recently obtained promising results applying CNNs for predicting the number of sessions crossing a network link (research work done in Wp4) we are encouraged to apply this type of deep neural networks in the domain of this use case.

8 References

[1] ONTIC. “Deliverable D4.2. Algorithms Description.” Internet: http://www.ict-ontic.eu/, Feb. 2015.

[2] ONTIC. “Deliverable D5.1. Use Case Requirements.” Internet: http://www.ict-ontic.eu/, Feb. 2014 [Dec. 1, 2015].

[3] ONTIC. “Deliverable D5.2. Progress on Use Cases.” Internet: http://www.ict-ontic.eu/, Feb. 2016 [March. 31, 2016].

[4] ONTIC. “Deliverable D5.4. Use Case #1: Network Intrusion Detection.” Internet: http://www.ict-ontic.eu/, Jan. 2017 [Jan. 28, 2017].

[5] ONTIC. “Deliverable D5.5. Use Case #2: Adaptive Quality of Experience Control.” Internet: http://www.ict-ontic.eu/, Jan. 2017 [Jan. 28, 2017].

[6] ONTIC. “Deliverable D5.6. Use Case #3: Proactive Congestion Detection and Control.” Internet: http://www.ict-ontic.eu/, Jan. 2017 [Jan. 28, 2017].

[7] Telecommunication Networks Group - Politecnico di Torino. “Tstat: TCP STatistic and Analysis Tool. 3.1.1.” http://tstat.polito.it/ (2016)

[8] Seufert, Michael, Florian Wamser, Pedro Casas, Ralf Irmer, Phuoc Tran-Gia, and Raimund Schatz. "YouTube QoE on mobile devices: Subjective analysis of classical vs. adaptive video streaming." In 2015 International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 43-48. IEEE, 2015.

[9] Casas, Pedro, Raimund Schatz, Florian Wamser, Michael Seufert, and Ralf Irmer. "Exploring QoE in Cellular Networks: How Much Bandwidth do you Need for Popular Smartphone Apps?." In Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges, pp. 13-18. ACM, 2015.

[10] Casas, Pedro, Michael Seufert, and Raimund Schatz. "YOUQMON: a system for on-line monitoring of YouTube QoE in operational 3G networks." ACM SIGMETRICS Performance Evaluation Review 41, no. 2 (2013): 44-46.

[11] Recommendation ITU-T "P. 910: Subjective video quality assessment methods for multimedia applications." International Telecommunication Union, Geneva (2008).

[12] Montresor, A., & Jelasity, M. (2009). “Peersim: A scalable p2p simulator.” In H. Schulzrinne, K. Aberer, & A. Datta (Eds.), Peer-to-Peer Computing (pp. 99-100). IEEE.

[13] Zegura, E. W., Calvert, K. L., & Bhattacharjee, S. (1996). “How to model an internetwork.” In INFOCOM (pp. 594-602).

Annex A QoE Degradation Detection

A.1 Data source management

Use Case #2 relies on capturing network traces and using them to detect QoE degradation situations. This task is carried out by the Network Trace Forwarder and is based on Tstat [1], which aggregates the actual network traces into flows. The output of Tstat is a set of CVS files where each row corresponds to a different flow and each column is associated to a specific observed measure in the flow. When useful, columns are grouped according to C2S - Client-to-Server and S2C - Server-to-Client traffic directions. A TCP flow is considered to start when the first SYN segment from client to server is observed, and is considered to be finished when either:

- the FIN/ACK or RST segments are observed;
- no data packet has been observed (from both sides) for a default timeout of 10 seconds after the opening SYN segment, or 5 min after the last data packet.

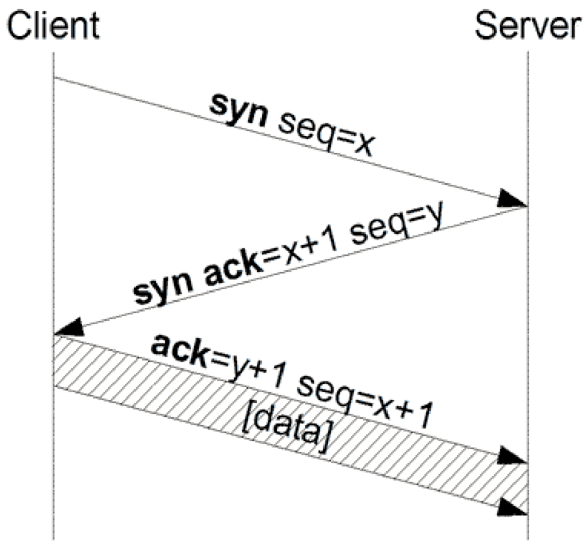


Figure 14: Diagram of the initialization (handshake) of a TCP connection[4]

Tstat discards all the connections for which the usual TCP three-way handshake is not properly observed. Then, in case a connection is correctly closed the flow information is stored in a file named `log_tcp_complete`, otherwise in `log_tcp_nocomplete`. If properly configured, Tstat can produce a `log_video_complete` file which logs every TCP video connection that has been tracked. In the UC #2 test-bed a basic trace forwarding procedure that mimics the way off-the-shelf DPI product work. In short, the Tstat output files are constantly monitored and every time a new row (a new record) is recorded, the trace information is forwarded to the Analytics Function.

The dataset provided by DellEMC comprises traces of 333 533 TCP sessions. In the next paragraphs, data cleaning and the feature selection procedure will be outlined.

Each recorded session is characterized by means of 144 tstat columns (features). There is one additional column which passes on the perceived QoE level (Good-Medium-Bad) of each session: `c_qoe_observed.205`.

[4] Caos, Tcp-handshake, CC BY-SA 3.0

In a first cleaning 11 rows were removed due to format errors, as not all fields were filled. In a second round of data cleaning 74 features were also removed. All these features carried the same value for all available sessions or are time-related values (the names and indices of every feature are those assigned by Tstat; only `c_qoe_observed.205` is a new feature, assigned after a process carried out by humans):

c_ip.1	c_ttl_max.51	c_win_scl.75	http_req_cnt.119
c_port.2	yt_itag.62	c_sack_opt.76	http_res.121
s_ip.15	yt_id16_46.61	c_mss.78	c_tls_SNI.124
c_pkts_ooo.12	s_ttl_min.57	c_pkts_fs.88	s_tls_SCN.125
s_port.16	vd_type_cont.59	c_pkts_reor.89	c_npnalpn.126
first.29	vd_type_pay.60	c_pkts_dup.90	s_npnalpn.127
last.30	s_ttl_max.58	c_pkts_unk.91	c_tls_sesid.128
c_first.32	vd_dur.63	c_pkts_fc.92	c_last_handshakeT.129
s_first.33	vd_rate_tot.64	c_pkts_unfs.94	s_last_handshakeT.130
c_last.34	vd_width.65	c_syn_retx.95	c_appdataT.131
s_last.35	vd_height.66	s_f1323_opt.96	c_appdataB.133
c_first_ack.36	yt_id11.67	s_tm_opt.97	s_appdataT.132
s_first_ack.37	yt_seek.68	s_win_scl.98	s_appdataB.134
c_isint.38	yt_red_mode.69	s_sack_opt.99	fqdn.135
s_isint.39	yt_red_cnt.70	s_mss.101	dns_rslv.136
c_iscrypto.40	yt_stream.72	s_win_min.105	req_tm.137
s_iscrypto.41	yt_mobile.71	s_win_0.106	res_tm.138
http_t.44	c_f1323_opt.73	s_pkts_fc.115	
c_ttl_min.50	c_tm_opt.74	s_syn_retx.118	

Table 2: Initial set of tstat features in UC#2

As a result of the data cleaning process, the following features are selected for analysis:

c_pkts_all.3	s_pkts_data.22	s_rtt_min.53	s_mss_max.102
c_rst_cnt.4	s_bytes_all.23	s_rtt_max.54	s_mss_min.103
c_ack_cnt.5	s_pkts_retx.24	s_rtt_std.55	s_win_max.104
c_ack_cnt_p.6	s_bytes_retx.25	s_rtt_cnt.56	s_cwin_max.107
c_bytes_uniq.7	s_pkts_ooo.26	c_sack_cnt.77	s_cwin_min.108
c_pkts_data.8	s_syn_cnt.27	c_mss_max.79	s_cwin_ini.109
c_bytes_all.9	s_fin_cnt.28	c_mss_min.80	s_pkts_rto.110
c_pkts_retx.10	durat.31	c_win_max.81	s_pkts_fs.111
c_bytes_retx.11	con_t.42	c_win_min.82	s_pkts_reor.112
c_syn_cnt.13	p2p_t.43	c_win_0.83	s_pkts_dup.113
c_fin_cnt.14	c_rtt_avg.45	c_cwin_max.84	s_pkts_unk.114
s_pkts_all.17	c_rtt_min.46	c_cwin_min.85	s_pkts_unrto.116
s_rst_cnt.18	c_rtt_max.47	c_cwin_ini.86	s_pkts_unfs.117
s_ack_cnt.19	c_rtt_std.48	c_pkts_rto.87	http_res_cnt.120
s_ack_cnt_p.20	c_rtt_cnt.49	c_pkts_unrto.93	c_pkts_push.122
s_bytes_uniq.21	s_rtt_avg.52	s_sack_cnt.100	s_pkts_push.123

Table 3: Set of UC#2 Tstat features after cleaning

In order to reduce the dimensionality of the dataset, a Principal Component Analysis (PCA) procedure is carried out (using the R language). This procedure is equivalent to the that of FreeScan. Once the dataset is cleaned and normalized, the 20 features with higher weights with regard to the first principal component are shown in the following table:

Variable	PCA1 Weight
0.2825937	s_pkts_all.17
0.2825766	s_ack_cnt.19
0.2441062	c_pkts_all.3
0.2441014	c_ack_cnt.5
0.2391385	s_pkts_unk.114
0.2354672	c_rtt_cnt.49
0.227788	s_ack_cnt_p.20
0.2262491	c_pkts_data.8
0.221656	c_bytes_all.9
0.2215922	c_bytes_uniq.7
0.1812075	c_ack_cnt_p.6
0.1743395	s_pkts_data.22
0.165843	s_bytes_all.23
0.1651281	s_bytes_uniq.21
0.1640106	s_rtt_cnt.56
0.1592224	s_pkts_push.115
0.1534585	s_cwin_max.99
0.1372518	c_cwin_max.76
0.1137737	c_mss_max.71
0.1121032	s_pkts_retx.24

Table 4: PCA1 Weights

The use of 20 variables is still too computationally costly and therefore, some domain knowledge will be used in order to get a manageable set of variables. The most important variable, according to the PCA analysis, is `s_pkts_all.17`. It logs the total number of packets observed from the server. If we consider this variable and the number of retransmitted segments (`s_pkts_retx.24`), also with a high weight with regard to PCA1, we can obtain the Packet Loss Rate.

`s_pkts_all.17` can be also used to compute the ratio of duplicated or out-of-sequence packets. It can be carried out considering `s_pkts_unk.114` (number of segments not in sequence or duplicate which are not classified as specific events as observed from the server), with a high weight with regard to PCA1 as well.

Another variable relevant to the domain is the bandwidth. It can be obtained as the ratio between the number of bytes transmitted in the payload, as observed from the server (`s_bytes_all.23`), with a high PCA weight as well, and the flow duration (`durat:31`).

Finally, `s_rtt_avg:52` is also selected. Although it is the 66th variable considering its PCA1 weight, is a relevant KPI when considering the domain. It records the network latency. That is, the average time taken for a packet for going from source to destination.

Actual analysis will consider a number of Key Performance Indicators, which are not used as such, as suggested by the state-of-the-art [8][9][10]. They are derived from some of the features of the dataset.

Variable	Description	Formula
Packet Loss Rate	Ratio between the number of retransmitted segments from the server and the total number of packets observed from the server	$\frac{s\_pkts\_retx.24}{s\_pkts\_all.17}$
Packet Rate	Ratio between the number of segments not in sequence or duplicated and the total number of packets observed from the server	$\frac{s\_pkts\_unk.114}{s\_pkts\_all.17}$
Latency	Average RTT computed by measuring the time elapsed between the data segment and the corresponding ACK	$s\_rtt\_avg{:}52$
Download Bandwidth	Ratio between the payload and the flow duration (from first to last packet)	$\frac{s\_bytes\_all.23}{durat{:}31}$

Table 5: QoE detection variables

A.2 QoE degradation detection

QoE degradation detection is carried out by continuously matching the network traces against a clustering model trained with the Dell EMC dataset over appropriate windows and with given thresholds to decide the degradation has started/ended. For a given location, a numeric value, `video_qoe`. is computed. For each new network trace, distance to the cluster centroids is computed and a cluster is assigned. As the clusters have been given a numeric score as well, the flow is therefore assigned a QoE score. The average of the flows score over a rolling windows is computed and whenever said average goes bellow a given threshold, an alarm is issued. If `video_qoe` goes above the threshold for a sustained time, the alarm is cancelled.

Two separate processes are used to detect QoE degradation: the training of the model and the application of said model to incoming network traces.

In order to **train the model**, the implementation performs feature selection over the training dataset. Only `s_bytes_all.23`, `durat.31`, `s_pkts_retx.24`, `s_pkts_all.17`, `s_pkts_unk.114`, `s_rtt_avg.52`, and `c_qoe_observed.205` are extracted. Next, data cleaning, normalization, and computation of the four new features is carried out according to the formulae in the previous section. See code in Python 2.7 with pandas:

```
# training dataset load and filtering
columns = ['s_pkts_retx.24', 's_pkts_all.17', 's_pkts_unk.114', 's_rtt_avg.52',
's_bytes_all.23', 'durat.31', 'c_qoe_observed.205']
df = pd.read_csv("log_video_complete_all.csv", sep = "," , usecols=columns)

# cleaning
df = df[(df['durat.31'].isnull() == False) &
        (df['s_pkts_all.17'].isnull() == False) &
        (df['s_pkts_unk.114'].isnull() == False) &
        (df['s_pkts_unk.114'] != 0.0) &
        (df['s_rtt_avg.52'] < 100.0)
       ]

# KPI extraction and generation of new dataframe
kpi_df = pd.DataFrame()
kpi_df['download_bw']      = df['s_bytes_all.23'] / df['durat.31']
kpi_df['packet_loss_rate'] = df['s_pkts_retx.24'] / df['s_pkts_all.17']
kpi_df['packet_rate']      = df['s_pkts_unk.114'] / df['s_pkts_all.17']
kpi_df['latency']          = df['s_rtt_avg.52']
```

```
kpi_df['labels']          = df['c_qoe_observed.205']

# normalization
kpi_df['download_bw'] = kpi_df['download_bw'] / kpi_df['download_bw'].max()
kpi_df['latency']     = kpi_df['Latency'] / kpi_df['Latency'].max()
```

K-means clustering is finally carried out on the new dataset (considering only the four new features defined in previous section; `labels` is not considered in the clustering procedure). The Elbow method is used to find the K in the K-means implementation. Five is the optimal figure. In the figures above, two cluster plots (with two features each) are shown. The first one shows the clustering just plotting the download bandwidth and the latency; the second one shows the download bandwidth and the packet rate:

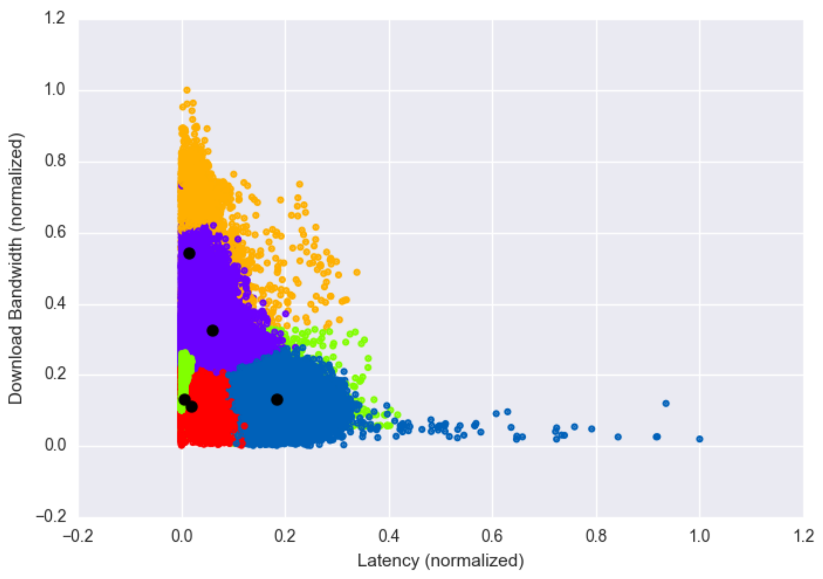


Figure 15: Clustering (download bandwidth vs. latency)

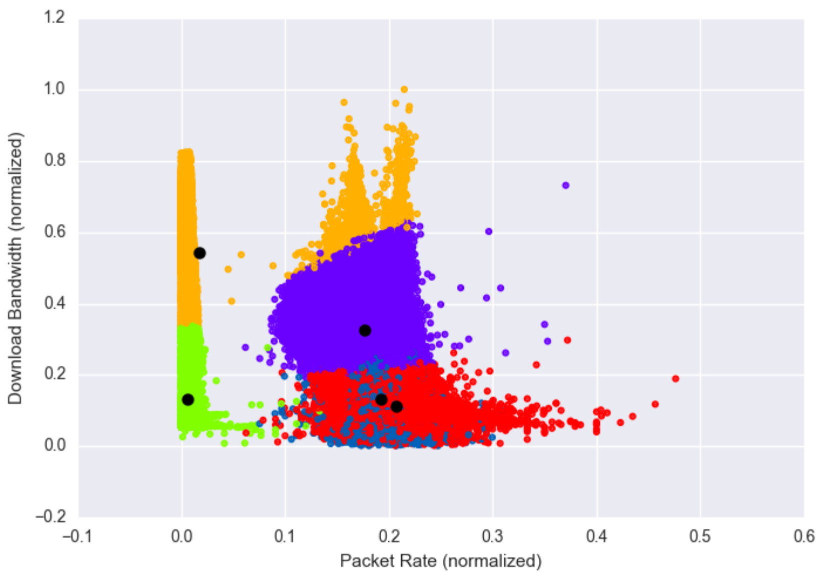


Figure 16: Clustering (download bandwidth vs. packet rate)

Once the clusters have been obtained, the share of each label is computed in order to determine whether the clustering is consistent with the perceived quality of experience. The results are shown in the figure below:

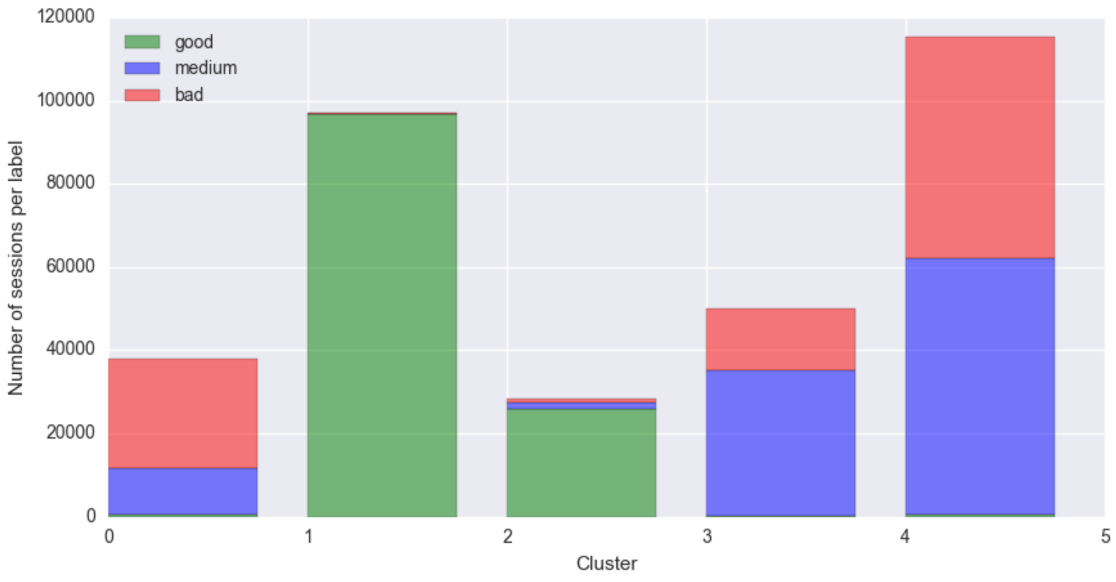


Figure 17: UC#2 Perceived quality per cluster

Such results are consistent as no cluster combines large shares of mutually incompatible labels (for instance, no cluster mixes large good and bad labels). Each cluster is assigned a different QoE numeric value, according to the levels outlined in section **¡Error! No se encuentra el origen de la referencia.**

Cluster	Label "good"	Label "medium"	Label "bad"	Score
0	1.22	29.52	69.27	3.96
1	99.71	0.01	0.29	8.98
2	91.03	5.76	3.21	8.63
3	0.01	69.98	30.01	5.10
4	0.46	53.47	46.07	4.63

Table 6: Per cluster score

The outcome of the model training will be a set of five cluster centroids and the score associated to each cluster, in a 0-to-10 scale. The **application of the model** is straightforward. Whenever a new trace arrives, the distance to the centroids is computed and the trace is assigned to the closest centroid. Depending on the cluster, a numeric QoE score is assigned to every session. QoE degradation detection relies on a rolling computation of the average QoE KPI, named `video_qoe`.

The computation of `video_qoe` is done in 30-second windows (`window_size`: 30 seconds). Computation is repeated every 15 seconds (`reevaluation_period`: 15 seconds), considering the last 30-second window. When `video_qoe` goes below 60% (`video_qoe_threshold`: 60%; `recovery_video_qoe_treshold`: 75%; `video_qoe_best_value`: 100%; `video_qoe_worst_value`: 0%) for more than 30 seconds (that is, one window), a degradation report is delivered to the PGF (degradation_persistence: 60 seconds) using the POST method. A `reportID` is computed for every report. The PGF sends back a session ID, as part of the `Location` header, that must be recorded.

The degradation report includes a `validity` element. Its default value is 60 seconds (`default_validity`: 60 seconds). It is handled in the following way: `start_time` is current time (epoch in milliseconds); `end_time` is current time plus 60 seconds (60000 milliseconds).

Computation goes on for the window time. Two situations can be found:

- video_qoe value recovers when it goes over 75% (recover_video_qoe_treshold), following a hysteretic cycle, to avoid transient states. If so, a cancellation must be sent by sending an update message (that is, using the same session resource, but using a PUT method). The validity period must last 0 second. That is, start_time and end_time are set to Epoch current time.
- video_qoe value does not go above the recover_video_qoe_treshold. If the degradation situation remains for the 11 subsequent computation, an update (PUT) is sent, with the same validity period (60 seconds).

To summarize, the following parameters are used:

Parameter	**Value**
window_size	30 seconds
reevaluation_period	15 seconds
video_qoe_threshold	60%
recovery_video_qoe_threshold	75%
video_qoe_best_value	100%
video_qoe_worst_value	0%
degradation_persistence	60 seconds
default_validity	60 seconds