mPlane

an Intelligent Measurement Plane for Future Network and Application Management

ICT FP7-318627

Demonstration Plan

Author(s):	Author names
FW (ed.)	A. Fregosi, A. E. Kahveci, E. Kowallik, G. P. Mattellini, C. Meregalli, S. Raffaglio M. L. Russo, A. Sannino, M. Scarpino
POLITO	S. Traverso
FUB	E. Tego, F. Matera
ALBLF	Z. Ben Houidi
EURECOM	M. Milanesio
NEC	M. Dusi
TI	F. Invernizzi
TID	I. Leontiadis, L. Baltrunas, Y. Grunenburger
NETvisor	Á. Bakay, B. Szabó, G. Rózsa
FTW	P. Casas

Document Number:	D6.1
Revision:	1.0
Revision Date:	16 Feb 2015
Deliverable Type:	DEM
Due Date of Delivery:	31 Jan 2015
Actual Date of Delivery:	16 Feb 2015
Nature of the Deliverable:	(R)eport
Dissemination Level:	Public

Abstract:

This deliverable describes the design specification of the demo architecture and the organization of the demonstration for the specific use cases that are defined in WP1. The document also provides a complete description for the selected use-cases that includes the demo strategy, mapping of components and impairments into the demo architecture. Finally, for each use case involved, the demo execution flowchart specifies the process that would be used as a baseline for the demo.

SEVENTH FRAMEWORK PROGRAMME

Contents

1 Introduction

The main objective of WP6 is to show the capabilities of the mPlane platform based on the demonstration of a selected subset of functionality with respect to the use cases defined in WP1.

Each use case consists of a list of probes (WP2), the collection of data through probes (WP3) and finally the reasoning analyses based on the collected data (WP4). The goal of the demonstration is to put together the needed components and enable the showcase of the selected use cases.

In Chapter 2, firstly we provide a description of the architectural design and specification of the demo infrastructure that will host the needed components of mPlane. Following, in Chapter 3 we provide the definition of the infrastructure and the description of the demonstration of each use case.

For each use case, the demonstration description is split in four parts:

- Demo strategy: The targets of the demo and the objectives to demonstrate (e.g.: pre-post service degradation and troubleshooting).
- Components mapping into the demo architecture: The positions of the components (e.g.: probes, repositories) within the demo architecture.
- Vantage point(s) mapping: The details of the probes used in the use case demonstration.
- Demo execution flowchart: The demo workflow specifying the interactions between components, algorithms and expected results to demonstrate.

2 The mPlane demo infrastructure

2.1 Introduction

Demonstration infrastructure and the implementation of the mPlane platform into the network are described in the following section.

2.2 Demo implementation of mPlane architecture

The demo implementation of mPlane architecture is made up of three main environments: test-plant network, Production POP network and mPlane Core network.
Testplant and mPlane Core environments are located in FASTWEB network and are directly inter-connected with TI-Lab mPlane network of Telecom Italia; mPlane Core network is physically located in the FASTWEB testplant Labs.
The architecture has been designed to meet all the technical requirements of the Use Cases to be demoed.

2.2.1 Testplant POP Network

FASTWEB testplant is a controlled testing environment designed to simulate a general POP network architecture with production-like access topologies and technology components.
Testplant mPlane demo architecture is outlined in figure 1 on page 7.
The architecture consists of the following elements:

- N°2 residential access lines: an ADSL2+ line with physical datarate of 20Mbs_1Mbs (TP-CPE-Router1) and a VDSL2 line with physical datarate of 100Mbs_30Mbs (TP-CPE-Router2). Each of the access lines is terminated at the customer side on a CPE (Customer Promises Equipment) provided with LAN routing capabilities and connected toward Internet with IP public address.
- N° 1 internal CDN network made up of a content Media Server connected (TP-CDN-MediaServer) to a fiber optic terminated dedicated router (TP-CDN-Router) connected toward Internet with IP public address.
- N° 3 servers (TP-IMP-Server1, 2, 3) capable of generating user defined network impairments at both L1, L2 and L3 traffic.

The architecture has been designed to enable the general routing mostly generated by the customer from its premises to the Internet remote servers. Customer routers (TP-CPE-Router1, 2) are connected to xDSL access equipment (TP-AC-DSLAM1 for ADSL2+, TP-AC-DSLAM2, 3 for VDSL2) at the 'POP access level', access links are aggregated and routed by the routing equipment (TP-BB-Router1, 2) at the 'POP backbone level' and finally the interconnection with Internet and the internal CDN is managed by the edge router (TP-EDGE-Router).
The high level testplant demo architecture results in two distinct access chains connected to the Internet and to the controlled internal CDN. This allows the application of the selected network impairments at all the different levels of the topology.

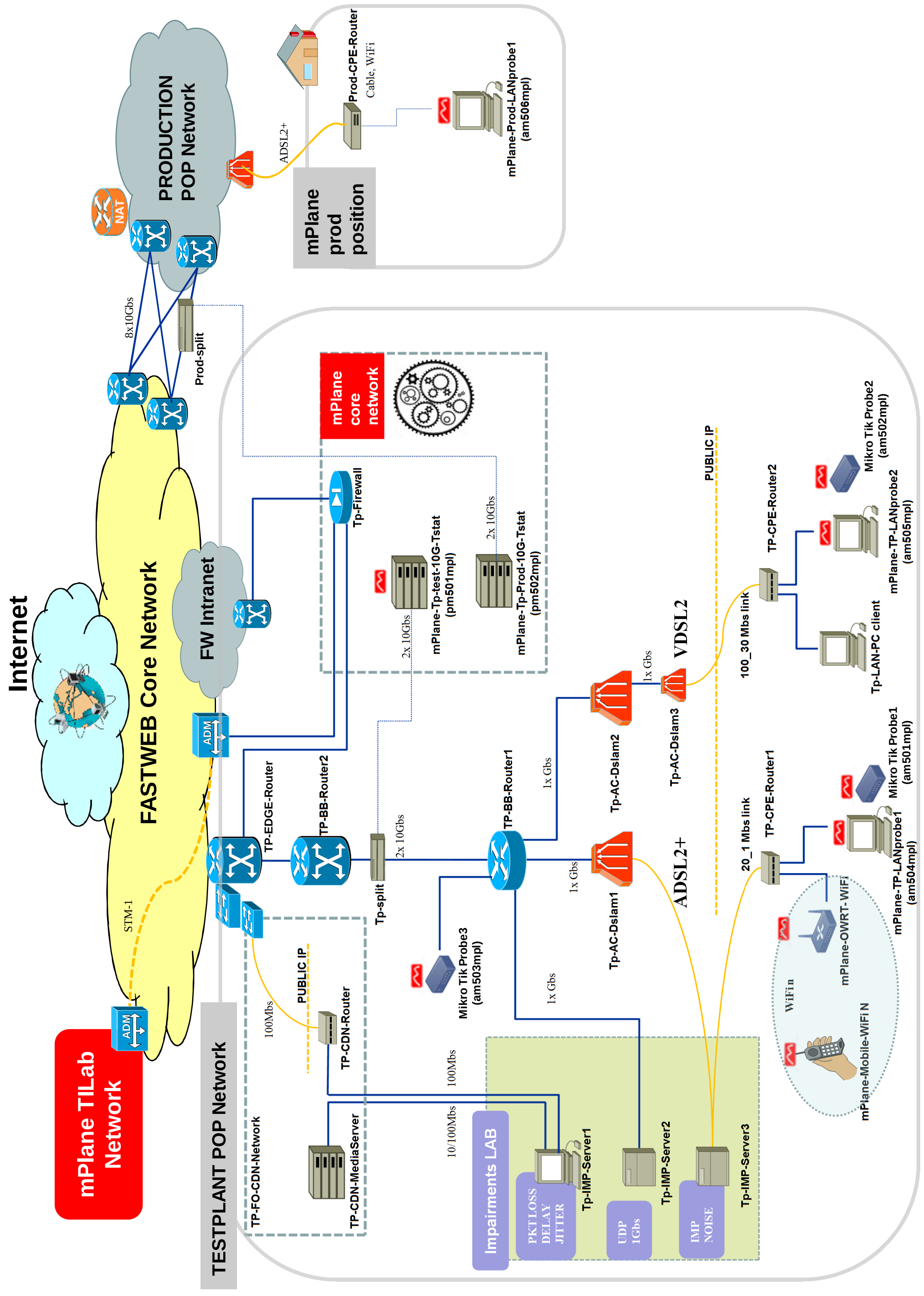

Figure 1: Testplant Demo network

2.2.2 Production POP network

During the demonstration phase live network traffic will be intercepted from a production FASTWEB POP in order to analyze the user experience of real customers.
A mPlane probe connected to a DSL line is positioned within the production POP. Moreover, for the demo purposes, a part of the real POP traffic will be monitored through two links that connect the POP with the interconnection network. Monitored traffic would flow through mPlane-Tstat probe that is a part of mPlane core network.

2.2.3 mPlane Core network

The mPlane core network consists of a dedicated network architecture designed to host servers implementing mPlane core components: Supervisor, Reasoner, Repository and WEB GUI. This network is provided with internal connections to Tstat Passive Probes and with external connections to the Internet, FASTWEB Intranet and TILab mPlane testplant. General architecture is outlined in Figure 2.

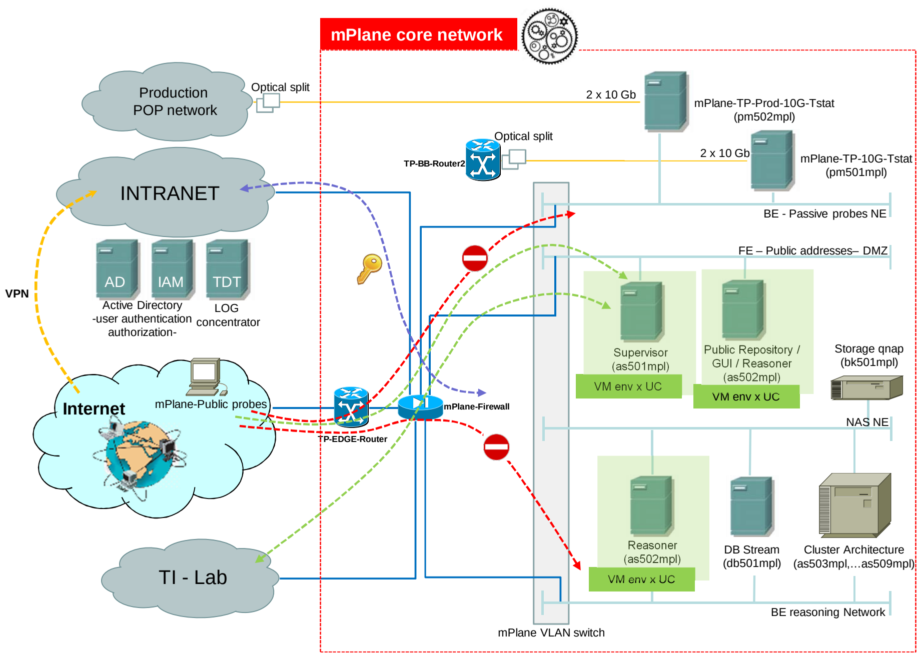

Figure 2: mPlane demo core network

The architecture consists of four networks obtained by VLAN segmentation of a Gigabit switch (mPlane VLAN switch), each of them is dedicated to a different class of equipment:

- 'BE - Passive Probes NE', back-end network hosting the two passive Tstat probes servers.

- ‘BE - reasoning NE’, back-end network hosting Reasoner and private Repository.
- ‘FE - Public addresses - DMZ’, front-end network with public addresses hosting Supervisor(s), public Repository and WEB GUI servers.
- ‘BE - NAS - NE’, back-end network hosting the storage server for data archiving.

Servers hosting Supervisor, Reasoner and public repository/GUI elements are provided with Virtual Machine environments in order to allow multiple UC software installations. On the frontend network it has been provided a Public Repository server for UCs requiring public probes to send measurements results directly to the Repository element.
The four VLANs are connected with each other and to the external networks by means of a dedicated firewall (mPlane Firewall) implementing specific security policies: the frontend network is a public addressed DMZ externally accessible from the Internet and TILab mPlane testplant, the backend networks are private addressed networks externally accessible from the Intranet FASTWEB network. Firewall FORWARD and DENY policies for the frontend network have been set up in order to grant minimum permissions necessary to enable mPlane communication. Internal connection among frontend and backend networks is not limited.
User Access to mPlane core network for operations and maintenance activities is only allowed when passing through the FASTWEB Intranet network submitting to FASTWEB authentication process.

2.2.4 Interconnection between FASTWEB and TI-Lab testplants

FASTWEB and TI-Lab testplants have been connected through one optical STM-1 link (155 Mb) which enables the communication between mPlane components within the two testplants. On the FASTWEB side, an ADM equipment terminates the STM link to the mPlane firewall through 1Gbs Ethernet link as shown in figure 1.
The routing between the two testplants can be configured in order to allow components on the TI-Lab network to exploit the FASTWEB mPlane core network features. Referring to figure 2, the TI-Lab mPlane network will be configured on the mPlane firewall as a backend network segment.

2.2.5 Vantage points

There are several vantage points available for the demo measurement purposes within the testplant POP network and the production POP network. Other vantage points can be placed within the TI-Lab network through the interconnection established between the two testplants.
The active and hybrid probes are able to perform measurements towards Internet and the internal Media Server. These probes can be located:

- within the testplant POP network, at the ‘access level’ behind the Customer routers (TP-CPE-Router1, 2) terminating the two available access lines (ADSL2+ and VDSL2);
- within the testplant POP network, at the ‘backbone level’ directly connected to the TP-BB-Router1. From this vantage point, the measurements performed exclude the ‘access level’ side of the network;
- within the production POP network, at the ‘access level’ behind the Prod-CPE-Router.

The two mPlane-Tstat passive probes are able to collect traffic data on the monitored links. These probes have been located:

- within the testplant POP network, at the 'backbone level' between the TP-BB-Router1 and the TP-BB-Router2. This probe intercepts the whole access network traffic;
- on the interconnection link between the production POP network and the Internet. This probe intercepts two links, corresponding to a portion of the whole POP traffic.

2.2.6 Impairments

Three different impairment equipment are available within the testplant network.
The first equipment (Tp-IMP-Server1) could be used to apply jitter, delay and packet loss at IP layer on the link that serves the Media Server.
The second equipment (Tp-IMP-Server2) is a traffic generator that generates traffic flows from a source IP to a destination IP. This impairment could be used to simulate traffic congestion toward Media Server and CPEs.
The last equipment (Tp-IMP-Server3) is a noise generator that can produce different noises, as defined by the ETSI standards, and apply them on the DSL copper line. The noise level can be tuned to cause packet loss or retraining on the DSL line.

2.2.7 Security aspects, data protection and privacy requirements

As depicted in figure 2, users can access mPlane core network for operations and maintenance purposes only by authenticating to the FASTWEB mPlane VPN. Due to FASTWEB internal policy, access to mPlane FASTWEB VPN is only permitted using RSA token plus a 8 digit PIN. If for the setup of the Use Case you need to access such VPN, you must submit a request to FASTWEB mPlane project Coordinator at least four weeks in advance. Users and devices installed in FASTWEB mPlane Core Network must also comply to all regulatory aspects and guidelines regarding user data protection, included those specified in deliverable D1.2.

3 Use cases demonstration

3.1 Estimating content and service popularity for network optimization

The goal of this use case is to detect which early contents will receive attention, so to proactively place the most-popular contents in caches closer to end users. The advantages comprise improving the overall Quality of Experience for the users and optimize the usage of network resources for the providers.

3.1.1 Demo strategy

The plan is to demonstrate that we can estimate the popularity of contents (i.e., YouTube videos) in the future, given the requests that the content has received at a given point in time.

We focus on a portion of the network that is served by one cache and we show how the prediction of popular contents and the views that a given content is going to have in the future have an impact on the cache utilization.

Targets to demonstrate:

- show that mPlane probes can successfully extract and parse users' requests for content on an actual network and aggregate the requests for a given content into time-series;
- show that mPlane analysis modules can effectively compare such time-series against pre-built models of patterns of growth and detect the future popularity within a pre-determined temporal horizon;
- show that the prediction of mPlane analysis module is accurate, by comparing the expected popularity of contents with the actual popularity of that content in the pre-determined temporal horizon;
- show that the promptly estimation of future content popularity is beneficial for a mPlane reasoner for optimizing the choice of which contents should be stored into a cache.

3.1.2 Components mapping into the demo architecture

The demo of this use case will include components from three work packages: WP2, WP3 and WP4.

Within the testplant PoP Network, we plan to monitor the users' requests of content generated from the testplant PoPs towards the Internet. A Tstat-based probe located into the mPlane core architecture will extract URLs and collect them in tiny time windows, e.g., minutes. The probe sends aggregated pairs of (URL, #requests) to the repository, which saves them on a database: at this point, data can be aggregated over a more course-grained time window, e.g., day(s).

Periodically, the mPlane analysis module will request the time series of how a given content evolves over time to the repository and it will predict the future popularity of contents in a given time horizon. Through the content popularity prediction algorithm, the analysis module returns the list of content popularity predictions. We will then compare the predicted popularity with the actual popularity of the content as the time goes by.

The mPlane reasoner will get the list of popular contents at a given PoP (as seen by the probe) and apply a caching policy algorithm which accounts for future popularity to decide which contents should be placed in the cache. The reasoner will also instruct the cache to periodically refresh and retrieve its contents, without the need of iterations.

3.1.3 Vantage points mapping

All the components will be placed into the mPlane core part of the testplant, and the probes will receive traffic from two main points of the network: the TI Lab and the actual PoP Network serving actual customers. The analysis algorithm will run separately on the data received from the probes, as it will hypothetically instruct two independent caches.

3.1.4 Demo execution flowchart

Figure 3 outlines the demo flow chart for the use case.

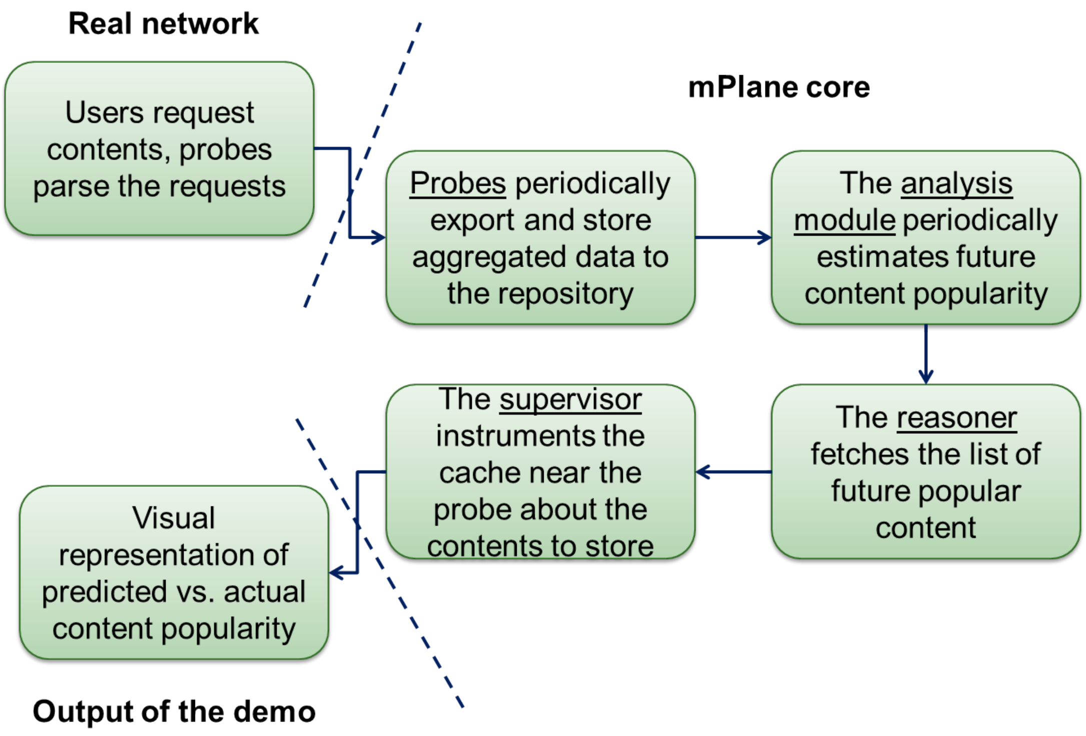
Real network
Users request contents, probes parse the requests
mPlane core
Probes periodically export and store aggregated data to the repository
The analysis module periodically estimates future content popularity
The reasoner fetches the list of future popular content
The supervisor instruments the cache near the probe about the contents to store
Visual representation of predicted vs. actual content popularity
Output of the demo

Figure 3: Demo execution flowchart of the use case ``Estimating content and service popularity for network optimization''.

3.2 Passive content curation

The content curation use case aims to provide a service that helps users identifying relevant content in the web. This use case monitors one or more probes in the network, detects URL clicks (called *user-URLs*) out of the streams of HTTP logs observed on the probes, and performs some analysis on these clicks in order to pinpoint the set of URLs that are worth recommending to users.

In D3.3 [2], we enhanced our user-URLs detection heuristics and modified them to run online at high rates of HTTP requests (up to 5 million per hour). Since this algorithm needs to run continuously on HTTP logs streamed from the mPlane probe, we decided to make it a scalable data-analysis algorithm that runs on the repository instead of an analysis module on its own. The output of this algorithm will be user-URLs, together with their timestamps, referrer and a flag saying whether they contain a social plugin or not.

In D4.2 [1], we sketched how we modified the structure of the analysis modules to work online, and we present in this document two new analysis modules which rely on the output provided by WP3: (1) the content versus Portal module and (2) the content promotion module.

Fig. 4 presents an overview of The passive Content Curation system. The passive Content Curation system takes as input HTTP requests from a raw data extraction module and outputs a list of sorted URLs to a presentation module, which can be a website or a mobile app. We briefly describe the high-level architecture of the passive content curation service, we will describe the details of the modules that compose The passive Content Curation system in the following section.
Raw data extraction is a set of traffic monitors running within the ISP network. Each monitor observes packets traversing a router and extracts HTTP requests, producing an HTTP log. Here, we use Tstat for extracting HTTP requests. We are interested in the following information from HTTP requests: *timestamp*, *URL*, *anonymized user id* (used to provide privacy through k-anonymity) as well as the *referer* and *user-agent* fields. All fields are extracted from the HTTP GET requests only, so that we do not need to access HTTP responses.

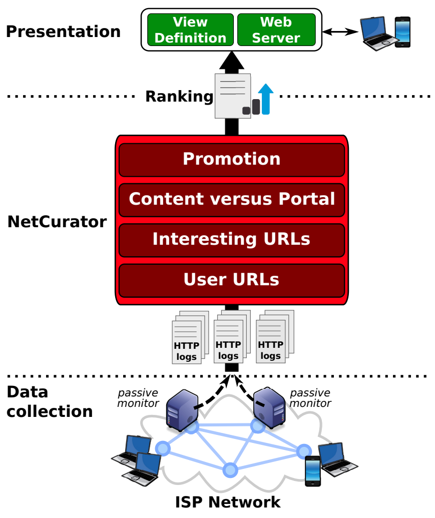

Figure 4: An overview of the content curation system.

The passive Content Curation system consists of four sub-blocks as depicted in Fig. 4. The *user-URL filter* identifies the HTTP requests corresponding to actual users' clicks. It eliminates the vast majority of HTTP requests that the browser automatically generates. The *interesting-URL* and the *content versus portal* module together select the set of user-URLs that are worth sharing with other

users. Finally, the *promotion* module takes as input this set of detected URLs (together with their timestamp and referer) and decides which ones to output to the presentation module.
Presentation is similar to news aggregation and curation services. It takes as input the list of promoted URLs and presents them in a user-friendly web portal (similarly to Reddit).

The purpose of this demo is to show that the mPlane architecture can be employed for services different from automatic network/QoE troubleshooting and optimization. This use case firstly aims to extract popular web pages delivered by the network, and, secondly, to promote those contents that may capture the interest of many. We remark that the whole content promotion system is designed not to offend users' privacy.

3.2.1 Demo strategy

This demo will rely on one of the passive probes Tstat installed at FW test plant network to extract interesting URLs, and present and collect (``cure'') them in a website.

By feeding the online algorithms described with a live stream of HTTP requests that will be made available in the test-plant, we will deploy a complete prototype running in the actual network.

More in details, we will rely on one of the Tstat probes available in the test-plant (i.e., at one of PoPs in FW network), which will monitor the browsing activity of the crowd of users. The mPlane proxy for Tstat will offer the capability of asynchronously exporting the HTTP logs generated by Tstat in a streaming fashion to a backend server running the algorithms for the extraction of user-URLs, first, and for the promotion of interesting-URLs, next. Such machine will run a mPlane-compliant proxy capable of accepting incoming streams of data from any passive probe. Once the setup will be ready and running, the presentation module, which acts as the backend of the website, will start selecting the web content to be presented in the website.

The website consists of three tabs, one for each promotion method. Each tab contains a content feed whose design is inspired by the ``wall'' implemented in popular social networks such as Facebook and Twitter, and URLs that make it to the feed are presented with a preview image, a title, and a description when available.

The promotion module of The passive Content Curation system takes as an input content-URLs. Since we aim to use The passive Content Curation system as the backend engine of a website, we test three different promotion mechanisms to use for as many sections (or tabs) in the website.

Hot. This mechanism is an adaptation of Reddit's Hot ranking algorithm [4], which promotes URLs that are both popular and recent. The algorithm behind Reddit's Hot ranking assigns each URL a score based on users' votes. We replace such votes the number of visits, and modify Reddit's formula to obtain the following:

$$Score = log(N_{views}) + \frac{T_{first} - T_{start}}{T_P}$$

N_{views} reports the number of views, T_{first} is the time corresponding to the first time the content-URL has been observed (i.e., visited), and T_{start} is the time corresponding to an absolute reference, i.e., the start of The passive Content Curation system. Finally, T_P represents the normalization factor that we use to define a ``freshness period'', and that we set to 12 hours. Intuitively, this formula finds a balance between the content popularity (the number of views) and its freshness (its age with respect to the absolute reference). When a content-URL stops getting attention, its ranking starts decreasing due to its age.

Top. This mechanism produces a simple ranking of URLs depending on the number of views. In the current version, the ranking can be provided for the last day, week and month.

Live news stream This mechanism focuses on one category of content-URLs, namely news, and aims to promote the freshest news seen in the network. In order to detect if a content-URL corresponds to a news, we rely on a predefined list of traditional news media websites in Italy: if the hostname of a content-URL belongs to this list, and if the content-URL has never been seen before, we tag it as news and promote it as fresh news. We construct the list of news websites by observing Google News, a popular news aggregation and indexing system that uses an active approach based on web-crawlers to collect and promote news. More precisely, Google servers regularly query a predefined set of popular news portals, looking for new articles to push on the front page [3]. To construct the list of news portals, we crawled the Google News front page every 20 minutes for a period of one week, looking for new websites. This allowed us to obtain a list containing more than 500 distinct news websites.

Finally, whenever a content-URL receives a visit, The passive Content Curation system updates the score of the URL and its number of views. The passive Content Curation system periodically recomputes the ranking and updates, if necessary, the database used to store the contents in the Hot and Top categories.

3.2.2 Components mapping into the demo architecture

The demo for this use case consists of four components:

- **Probe**: As said above, this demo will rely on Tstat and its mPlane-compliant proxy to feed the content promotion platform. Tstat will be installed at the PoP of the test plant.
- **Repository**: The content curation platform does not require any sophisticated repository to run, as the content extraction and promotion algorithms processes the stream of HTTP requests on-the-fly, without the need of storing any information.
- **Reasoner**: As described in D4.3 the content curation platform does not implement any ``reasoning'' or iterative procedure. The content extraction and promotion algorithms process the stream of HTTP requests without the need of running any further measurement. These modules (the red blocks in Fig.4), together with the presentation module, i.e., the website, will run on a server in the lab of the test plant. This server will run a mPlane-compliant proxy to receive the HTTP stream from the probe.
- **Supervisor**: the Supervisor will be hosted by a third machine in the test-plant, and it will be responsible to register and authenticate the other components, and to start and stop the asynchronous exporting of data from the probe to the machine hosting the content curation platform.

3.2.3 Vantage points mapping

This demo will rely on the passive probe 'mPlane-TP-Prod-10G-Tstat' available in the test plant.

3.2.4 Demo execution flowchart

The execution of the demo will be very simple:

1. Run the supervisor.
2. Register one of the available passive (Tstat) proxy and get the list of its capabilities.
3. Register the machine running the content curation website and get the list of its capabilities.
4. Send specifications to the content curation platform and to the probe to establish an asynchronous export of the HTTP logs in a streaming fashion
5. Observe the live feed of contents in the website

3.3 Active measurements for multimedia content delivery

The multimedia content delivery UC aims to provide a mechanism for service assurance for providers (telecommunication AND content providers) and also a tool for end customers to get an early notification and analysis of problems they may have encountered.

This mechanism is based on a set of coopearating probes from different technologies and a supervisor with reasoner that initiate an "investigation session" -- i.e. root cause analysis action--, once media delivery problems are detected, or such a problem is indicated by a user.

A principal goal of the Multimedia content delivery use case is to demonstrate the cooperation and orchestration of several probe types from various partners, i.e. prove that a compound of information delivered by probes that test and investigate a networking scenario from multiple aspects can efficiently be applied for the practical identification of typical network quality problem.

3.3.1 Demo strategy

The demostration will present various multimedia content sources (OTT servers) and various clients consuming such contents. It is shown that errors introduced at various points in the delivery chain cause errors or degradation in the service. The Reasoner being informed of the problem (i.e from probe measurements or through input of an external notification), will instruct the Supervisor to collect information from probes operating in the near past, and also initiate a detailed analysis by activating further measurements on other probes.

Collecting the information from these probe sources will gradually allow the probe to present a diagnosis of growing confidence level. Total analysis time is expected to range between 20-120 seconds in typical situations. The evolution of the diagnosis is constantly presented by the reasoner status, and, upon reaching a certain confidence threshold, notifications are also sent to affected clients. (These clients subscribe to patterns of notifications, allowing a selective notification for different parties).

The storyboard of the demostration is as follows:

- Demonstrating the normal service scenario, 2 content delivery servers (OTT servers) also representing 2 different CDN hubs will deliver multiple video content titles (some on-demand, some live contents), and 4-6 clients (mobile and fixed clients) will access these titles in an arbitrary pattern. Viewer experience is fair at all clients.

- The demo scenario will allow for simulation of various issues
  - Content missing from either or both CDN servers
  - Operational or performace problem at either server
  - Degradation of individual access lines, or degradation at the aggregated/core network segments that affect multiple viewers simultaneously.
- All clients in the scenario will include a QoE feature that reports quality problems to a central service assurance function.
- Some of these clients will also have probes available for remote operation by the "service assurance center" (SAC).
- Also, the network will contain QA devices, i.e. probes that are not bound to clients, but are available to be controlled remotely to run on-demand tests.
- All measurements will be driven by the SAC, which is indeed a Supervisor with an integrated Reasoner. The Reasoner will initiate the tests on the probes as needed, based on the previous test results, and also on a-priori information on the network's topology and on the services offered (i.e., the content titles available at either CD servers). The reasoner will use a rule-based inference engine to select the probes and parameterize the tests to be run next.

3.3.2 Components mapping into the demo architecture

- The following mPlane probes will be used in this UC demonstration.
  - OTT-probe (NETvisor)
  - BlockMon with integrated Tstat-probe (NEC/POLITO)
  - GLIMPSE (FHA)
  - MobileProbe (TID)
- Active CPE-side probes will be deployed on the OpenWrt-based MiniProbe framework/execution environment.
- Supervisor and Reasoner: the demonstration will include a single instance deployed in the demo SAC (co-located, i.e. on the same virtual machine). The Supervisor is based on the mPlane RI. The Reasoner architecture is TBD, probably implemented as a custom mPlane Client.
- Repositories MiniProbes (OTT probes) have integrated repositories which collect data on the probe (local data collection). Other modules will probably use one or possibly two repositories that are integrated with the center (central data collection).

3.3.3 Vantage points mapping

Probes are deployed in the following vantage points:

- Passive probes at the network exit point of each CD server/hub
- Passive probes on mobile devices (mobile probes)
- Active probes at various subscriber locations with local LANs (e.g. home networks or business networks of any size)
- Active OTT probes deployed on mobile test probes, i.e. probes with a mobile uplink (USB dongle)

3.3.4 Demo execution flowchart

1. SAC Reasoner receives notification of service degradation, via user feedback or from passive probes or from an active probe executing periodic service assurance test.
2. Reasoner activates measurements via Supervisor at various probes.
3. Probes execute measurements and return results.
4. Reasoner analyzes incoming results and updtates its inference status. Additionally, it may initiate further measurements (go back to #2), or publish diagnosis notifications of error to clients with subscriptions on the generated diagnosis types.

3.4 Quality of Experience for web browsing

3.4.1 Demo strategy

The Quality of Experience for web browsing use case aims to identify the root cause of a poor performance in a browsing session, that is, a high page load time during the session. As stated in Deliverable D1.1, several factors have impact on the page load time. We can divide these factors in groups based on the ''distance'' between the end user and the web server serving the web page. We can identify the following groups:

- **Local:**
  - the probe itself can be overloaded;
  - the Local Area Network where the probe is connected can be congested;
  - the Router/Gateway can be malfunctioning, misconfigured or overloaded;
- **Close:**
  - the ISP, identified by the first hops in a Traceroute towards different destinations;
- **Far:**

- the web server can be down, overloaded, under attack;
- the backbone network can have some intermediate hops overloaded.

The demonstration aims at showing how it is possible to get information on the root cause exploiting probe side (i.e., end user) measurements, both passive and active. We will highlight how basic (even if possibly incomplete) root cause analysis can be executed on a single probe, while a more complete and exhaustive picture will be obtained at the repository level, from the Reasoner.

The single Firelog probe is composed by four parts:

1. **Passive.** The core component of the probe is an instrumented headless browser, coupled with a modified Tstat probe. During a browsing session, objects from the web page are associated with the corresponding TCP flows collected by Tstat;
2. **Active.** After the browsing is completed, Ping and Traceroute are exploited for getting active measurements from the collected IP addresses;
3. **Diagnosis.** A diagnosis module runs on top of the collected data, providing information on the root cause;
4. **Exporter.** A Flume agent runs on the probe and sends collected data to a Hadoop File System (HDFS) repository for root cause analysis.

Optionally, impairments like packet loss on the gateway of the probe's LAN or probe's network congestion can be exploited for forcing the diagnosis algorithm to give different results.

3.4.2 Components mapping into the demo architecture

- **Probe**: the Probe will come as a standalone application which can be installed on every Linux laptop. It contains both the browsing module and the modified Tstat probe. We are working on a hardware probe which can be deployed in the test plant.
- **Repository**: for a first level demonstration, the repository is not exploited, as the local diagnosis is returned by the probe itself. Discussion is on going to install an OpenStack cluster on the test plant, and thus provide a demonstration also of the analysis modules within the repository.
- **Reasoner**: the Reasoner (if used) will be accessible through a server in the test plant, exposing the web interface for requesting the analysis.
- **Supervisor**: the Supervisor will run on a third machine in the test plant: it will send commands to the probe.

3.4.3 Vantage points mapping

Any machine at the LAN side (client side) can be used for installing the probe.

3.4.4 Demo execution flowchart

The demo will be executed as follows.

1. Start the supervisor, register the probe;
2. Send an URL as specification to the probe;
3. Receive the local diagnosis for that URL;
4. (optional) Browse to the web page of the Reasoner, insert the URL and get the diagnosis based on the data in the repository.

3.5 Mobile network performance issue cause analysis

3.5.1 Demo strategy

In a typical scenario where a user streams a video on a mobile device from a popular service like YouTube, a request to the content server is made to receive the video data. When the server receives the request, it sends the data either directly from the content server or through a Content Distribution Network (CDN). Then the data stream enters the Internet Backbone until it arrives to the client's ISP network. If the client is connected on a cellular network, then the data is delivered to the mobile from the client's serving cellular tower. If the device is connected from a Wi-Fi home network, then the video is delivered over a broadband access link to the home gateway and finally to the mobile device.

Each hop of the data path may suffer from impairments that can affect the smooth delivery of the video and therefore the user's experience. Congestion or band- width bottlenecks in the local or remote network segments, high load on the devices and problems in the wireless medium are some of the most significant issues that cumber the performance of video streaming services and contribute in the user's QoE degradation.

To detect the types of failures that may cause issues during the video playback, we need to place measurement probes at multiple vantage points (VP) so that we can extract performance metrics from different segments and devices along the path. In an ideal configuration, probes in all the intermediate devices of the path would provide us with measurements regarding the performance of each individual hop.

However, our current approach only requires probes at the mobile device, the home router and the content server. We only use these three points as they allow us to capture issues at the boundaries of each of the three important entities in the video delivery path, the user, the ISP and the content provider.

3.5.1.1 Setup

To demonstrate this approach we simulate a range of problematic scenarios in different segments of the data path that potentially cause interruptions in the playback and QoE degradation. An example of the setup is shown in Figure 5.

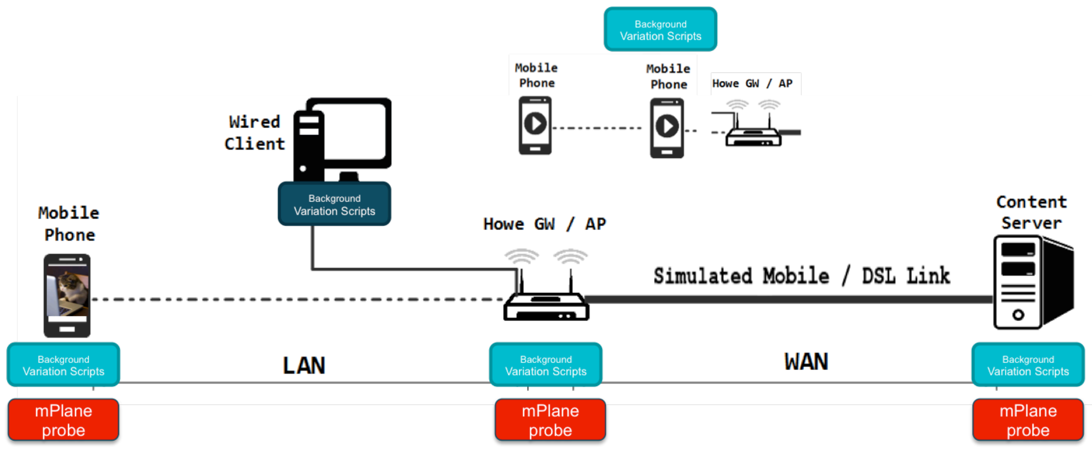
Background Variation Scripts
Mobile Phone
Mobile Phone
Howe GW / AP
Wired Client
Background Variation Scripts
Mobile Phone
Howe GW / AP
Content Server
Simulated Mobile / DSL Link
LAN
WAN
Background Variation Scripts
Background Variation Scripts
Background Variation Scripts
mPlane probe
mPlane probe
mPlane probe

Figure 5: Mobile probe setup

We set-up a simple testbed with a video server, a router/AP (Access Point) and one or more Android phones. The phones are connected to the Wireless LAN of the AP and the server is in turn connected via an Ethernet cable to the router. A wired client acting as a background traffic generator to introduce variability, is also connected to the router. We use tc and netem to simulate a DSL link by shaping the downstream of the link between the server and the router to simulate a mobile broadband link. It is possible that we are going to use the FW infrastructure to inject variability and problems.

The video server operates on Linux with the Apache HTTP server installed. The videos from the top 100 most viewed list have been previously downloaded from YouTube to the server in either Standard or High Definition to ensure the diversity of the collection. For the router/AP we used a Netgear WNDR3800 running OpenWRT. The access point of the device was configured to work on the 5GHz band in order to minimize interference from surrounding sources and we verified that no other devices were operating in the same frequency. For the mobile client, we used a Samsung Galaxy S II running Android 4.4.2. The mobile application that we developed is responsible for performing HTTP requests to the server and open the returned video stream using the default Android media player. As soon as the playback finishes the application repeats the process by launching another random video from the list.

3.5.1.2 Background variations

To generate the dataset for the controlled experiments, for each scenario we perform multiple iterations with random videos and gradually increment the intensity of the problem until we observe frequent rebuffering events. At the same time, attempting to create more realistic network conditions, we introduce variance to the system during every experiment by adding synthetic traffic workloads of different patterns and at random intervals. For that purpose, we use the Distributed Traffic Generator that supports traffic generation based on different applications such as Telnet, FTP, gaming, VoIP and more. We also use ApacheBench (ab) to create realistic HTTP traffic. Both tools were set to create traffic at random intervals and for random periods during the experiments. However, notice that background variations can be also introduced by the demo setup organizers and we will consider using their tools.

3.5.1.3 Simulation of problems

Apart from constant background variations, we generate a set of specific problems to label our measurements. Specifically, we create scenarios for the controlled experiments that represent real-world problems and we compile a list of common faults that we will simulate to cause stalling during the video playback. These scenarios can be grouped in three basic categories, networking, device hardware and wireless medium issues.

The generation of the problematic conditions can be performed in an automated fashion to easier control the experiments without the need of supervision. In more detail, the demo operator can select a possible problem to be generated and then a new experiment starts, the background traffic sources and sinks are dispatched. At the same time, the experiment controller script sets up the environmental parameters that are specific to the problematic scenario that we want to simulate. In the following step, a random video is selected and launched from the top 100 list. Based on the duration of the video a timeout is calculated so that we can detect when a video has finished playing plus some extra delay due to stalls and startup time. If the video has finished or timed out, we collect and aggregate the statistics from all the VPs for the given video session.

Shaping and Congestion. In the first category we have the LAN or WAN congestion and LAN or WAN shaping scenarios. These cases correspond to real-world conditions where the resources of the network are limited due to increased traffic, or due to bottlenecks such as slow links or bandwidth caps. To simulate LAN congestion, we use multiple iperf instances to transmit UDP traffic from the wired LAN client to the router, while for the WAN congestion we generate the traffic with the same method but from the server to the router. The traffic shaping scenarios are simulated with different bandwidth caps, delay and loss on the downstream of the related link. For the LAN shaping we select to limit the available bandwidth based on the data rates offered by common 802.11 standards such as a, b, g and n that are capable of providing rates per stream ranging from 1 up to 70Mbit/s. In these scenarios we apply delay with a normal distribution around 1ms and zero loss.

For the WAN traffic shaping we apply caps, delay and loss according to the distributions of the respective values observed in the measurements in the wild. For clients connected over mobile broadband we obtained throughput values normally distributed around 5.22Mbit/s, delay of 100ms and loss of 0.75

Mobile Load. In the second category we examine cases where the high load on the device hardware does not allow the proper decoding and playback of the video. This type of problems are more common on handheld devices that come with limited hardware capabilities. The hardware load simulation is performed with the Linux workload generator stress tool that allows CPU, I/O, memory and disk workload generation in order to stress-test the host system.

Low RSSI. In this scenario (low RSSI) we simulate poor signal reception by placing the phone far from the AP and blocking the line-of-sight with physical objects. As a result, there is degradation in the wireless link's SNR and the available data rate.

WiFi Interference. This scenario involves the creation of interference on the wireless channel from external sources. In real use cases, interference can be caused by adjacent devices transmitting or receiving on the same frequency range. For our experiments we create interference by generating large traffic workloads on a second WLAN where the AP operates on the same channel as the AP we use for our measurements.

3.5.2 Components mapping into the demo architecture

3.5.2.1 Repository

At the same time that we simulate a problem our probes record the network conditions. After each video flow finishes the data are gathered into a mPlane repository. The data are retrieved in a mongoDB database and can be retrieved using the mPlane Reference implementation if required.

The collection server can be collocated with the video delivery server to simplify the setup.

3.5.2.2 Reasoner

Afterwards we use machine learning methods to learn the correlations between performance and QoE metrics and to create a predictive model for detecting and characterizing the root cause of playback problems. For the data processing and analysis we use version 3.6.10 of Weka. Weka is a collection of machine learning algorithms and tools for processing, classification, regression and clustering. The server also has a training set store that helps to initially classify connection and it can later keep learning from future samples.

The machine that performs the machine learning can be collocated with the video server and/or the data collection server to simplify the setup.

3.5.2.3 Visualization (GUI)

The visualization is not yet implemented but we envision a simple tool to launch an experiment (i.e., select a specific problem and its severity). Afterwards the traffic generators will be launched to emulate the problematic network conditions. A video will then be launched from the mobile device.

At the end of a video the mPlane classifier will use the mPlane probe data to estimate the root cause and the location problem and the results will be visualized to the demo operator. Hopefully, the detected problem will match the one that was emulated.

3.5.3 Vantage points mapping

- The phones are connected to our access point (openWRT)
- The access point is connected to the ADSL/VDSL router.
- The video server can be near the FW Intranet or the Internet or the CDN.
- The repository can be anywhere in the FW network (preferably) or the Internet.
- The reasoner can be anywhere in the FW network or the Internet. Ideally it should be collocated with the repository.
- The GUI can be a laptop connected to the same access point with the phone (or any machine with a screen that has access to the reasoner).

3.5.4 Demo execution flowchart

i) Initially the demo operator selects an emulated problem, ii) the network conditions are adjusted to emulate the problem, iii) a video is started, iv) the mPlane probes collect data, v) when the video finishes the data are uploaded to the collection server, vi) machine learning is applied to detect the problem based on the network measurements, vii) the results are visualized.

3.6 Anomaly detection and root cause analysis in large-scale networks

This use case targets the continuous monitoring of large-scale network traffic, aiming to detect and diagnose anomalies potentially impacting a large number of users. The use case particularly focuses on the most popular web-based services (e.g., YouTube, Facebook, Google Services. etc.), delivered by complex network infrastructures maintained by omnipresent Over The Top (OTT) content providers and major Content Delivery Networks (CDNs) such as Google, Akamai, Limelight, SoftLayer, etc.. Detecting and diagnosing anomalies in such scenarios is extremely complex, due to the number of involved components or players in the end-to-end traffic delivery: the Content Provider, the CDN provider, the intermediate Autonomous Systems (ASes) of the transit Internet Service Providers (ISPs), the access ISP, and the terminals of the end-users. This high complexity motivates the usage of mPlane to improve the visibility on the traffic and on all the intermediate components. And more specifically, the diagnosis of the detected anomalies requires the coordinated guidance of the mPlane Reasoner, which shall decide the specific measurements and deeper analysis to perform, once an anomalous event is detected. In the demo we shall focus on the specific case of YouTube QoE-based traffic monitoring, detecting and diagnosing real anomalies occurring in the distribution of YouTube videos.

3.6.1 Demo strategy

The plan is to demonstrate that we can detect and provide troubleshooting support for real large scale anomalies occurring at the Internet level on web-based services. In particular, the demo shows how the mPlane can detect anomalies based on its Anomaly Detection modules, using both QoS-based and QoE-based performance metrics, as well as how the iterative analysis performed by the mPlane reasoner can correlate measurements from multiple probes and vantage points to understand the root causes of the detected anomalies. The following is a list of the targets to demonstrate:

Targets to demonstrate:

- show the interaction between passive and active measurements, using a common repository to store and analyze the monitored data.
- show that mPlane Anomaly Detection modules can effectively detect anomalous behaviors related to both QoS-based and QoE-based performance metrics.
- show how the mPlane Anomaly Detection modules can automatically learn and adapt to normal traffic variations to avoid false alarms and perform in a semi-autonomous fashion.

- show how the mPlane reasoner is capable of instructing new measurements on the fly when some specific events are detected.
- show the integration of external data sources within the mPlane framework, including both external databases and external active monitoring platforms (in particular, we consider RIPE Atlas as a distributed monitoring framework, based on active measurements).
- show how the mPlane can provide elaborated analysis of multiple measurements to potentially diagnose the root causes of the detected problems.
- show how the mPlane can rely on multiple Analysis Modules to enrich the traffic monitoring and analysis process.
- show how the mPlane can store and perform historical data analysis using its repositories to better support the analysis of relevant anomalies.
- show how the mPlane can learn new specific data models from stored (off-line) and streaming (on-line) data through machine learning approaches, particularly using the DBStream repository and analysis framework.

3.6.2 Components mapping into the demo architecture

The demo of this use case will include components from the three layers of the mPlane (i.e., measurement layer, large-scale data analysis and storage layer, and advanced analysis and reasoning layer). In particular, the following list details the different components per layer of the mPlane:

- **Probes**: at the measurement layer, we consider both passive and active measurements. Passive measurements are performed through a Tstat passive probe ("mPlane-TP-Prod-10G-Tstat" in the testplant), attached to the interconnection links toward the Internet of the production PoP Network (i.e., the monitored traffic is real customer traffic). Active measurements are performed through the RIPE Atlas monitoring framework, relying on the mPlane RIPE Atlas proxy to instantiate and coordinate the active measurements.

- **Repositories**: at the large-scale data analysis and storage layer, the use case demo relies in the DBStream repository (located at the "BE - reasoning NE" VLAN in the test plan) to store the passive and active measurements obtained from the aforementioned probes, as well as running the analysis algorithms which shall unveil the types of anomalies described before. The NAS storage server will also be considered as an additional data repository (located at the "BE - NAS - NE" VLAN in the testplant).

- **Analysis Modules, Reasoner and Supervisor**: the use case relies on the WP4 Anomaly Detection analysis module to detect on-line the targeted traffic anomalies, which is based on the analysis of the empirical distributions of several traffic features. A Reasoning system will be running at the "BE - reasoning NE" VLAN network, either on a separated machine or at the same hardware running DBStream. An instance of the mPlane Supervisor shall register all the components and will be deployed at the publicly addressed network ("FE - Public addresses - DMZ" VLAN in the testplant).

3.6.3 Vantage points mapping

All the described components will be deployed at the mPlane core network of the test plant, at the aforementioned locations. The Tstat probe monitors two 10Gbs links (2 x 2 link up/down). The Anomaly Detection analysis module runs directly on top of DBStream.

3.6.4 Demo execution flowchart

The demo of this use case is not meant to be performed on a live setting, but to let the complete mPlane system running for several months, collecting and analyzing YouTube measurements in the quest for anomalies. The complete workflow involves all the aforementioned components. The workflow is iterated and guided by the mPlane Reasoner, which is in charge of coordinating the analysis of the passive and active measurements and the results provided by the Anomaly Detection analysis modules.

Traffic is passively monitored at the production PoP network. Traffic is monitored at the flow-level, generating a large set of flow-statistics for all the downlink and uplink traffic. Using Tstat flow filtering and traffic classification capabilities, only flows related to YouTube videos are retained for further analysis. Some of these per-flow statistics include: flow size, flow duration, average download throughput, video bit rate, server IP, RTT, etc..

Flows captured at the passive probes are periodically exported to DBStream, which is in-charged of running the Anomaly Detection analysis modules. Tstat flow measurements are combined with two other types of measurements: (i) external data coming from geo-localization services such as MaxMind[1] and IP address analysis services such as Team Cymru Community Services[2], and (ii) inter-AS path performance measurements, generated through the usage of the geo-distributed active measurements framework provided by RIPE Atlas.

The main triggering event of an anomaly alarm is the detection of a QoE-relevant degradation event impacting a large number of YouTube users. Given that the Tstat flow measurements report the average flow download throughput as one of the monitoring KPIs, we rely on the QoE-based indicators presented in D4.3 to check whether download throughput is high enough to avoid stalling, or on the contrary, it is impacting the experience of the end-users.

The following list details the demo execution steps performed during the continuous monitoring:

- all traffic flows are analyzed by Tstat at the vantage point, and those belonging to YouTube are exported into DBStream.
- the anomaly detection algorithm runs continuously on the YouTube flows within DBStream, considering as KPIs the per-flow average download throughput (to detect performance issues) and the number of flows served per /24 CDN subnetwork (to detect Google cache selection changes).
- when an anomaly is detected as a major shift in the distribution of flows throughput towards lower throughput values, the diagnosis analysis is triggered. The diagnosis analysis is iterated by the Reasoner, following the diagnosis graph presented in D4.2.

[1] https://www.maxmind.com
[2] https://www.team-cymru.org/

- the first step is to verify if the detected anomaly is statistically consistent, i.e., that it is not caused because of a big drop in the number of samples considered in the empirical distribution computation.
- then the analysis verifies if this detected anomaly is actually impacting the QoE of the users, by analyzing the QoE-based KPIs as defined in D4.3.
- the first diagnosis event to verify is a main drop on the time-series related to the empirical entropy of the operative system type of the devices downloading the captured flows. A drop in the empirical entropy would flag a major concentration on the distribution of the OS type of the devices, indicating a possible relation to the OS type.
- the second event to verify is the occurrence of performance degradation in the corresponding end-to-end paths carrying the impaired YouTube flows. Events tracked on the time series related to packet re-transmissions, queuing delay, etc. are checked in order to identify path congestion.
- if path congestion is identified, then the Reasoner instructs active measurements from geo-distributed probes (e.g., using RIPE Atlas) to identify the specific AS or sub-path causing the performance degradation.
- if no path performance degradation is observed, the analysis checks for events related to load balancing and cache selection modifications in the Google CDN serving the YouTube flows.
- if no cache selection modification events are present in the logged events at the specific times of the detected YouTube QoE-based anomalies, the drilling-down checks for the occurrence of inter-AS routing changes which might be linked to the detected anomalies.
- if cache selection modifications are present, then the analysis focuses on understanding if the new selected servers are the origin of the problems. For doing so, different application-level KPIs are verified on top of the monitored traffic, such as server elaboration times, TCP flags, etc...

After running these steps over a long period of time, the demo will allow to present the obtained results through a direct query of the mPlane framework. We expect that some major anomalies will be captured during the time span of the demo traffic monitoring.

3.7 Verification and Certification of Service Level Agreements

The SLA verification and certification aims to measure SLA in terms of:

- Two way delay, measured using ICMP.
- Throughput as seen by TCP.
- Capacity as seen by UDP.

These will be done using the mSLAcert probe, that has this three services integrated and is mPlane compliant.

3.7.1 Demo strategy

The main targets to demonstrate are listed below:

- deliver the service in normal network conditions to show the expected quality. In this case, RTT, TCP throughput and UDP capacity measurements will be run on normal network conditions. Possibly the path that will be tested will not have heterogeneous traffic on its nodes, so most of the traffic will be from the test of the SLA.

- deliver the service in presence of specific impairments to show service degradation. We will introduce high delay and jitter on the network. The reasoner will detect the bandwidth degradation by running continuous tests.

- show how mPlane detects the service degradation. The reasoner will compare the measurements run over the normal conditions and the degraded network conditions. It will recognize variations in RTT, TCP and UDP throughput.

- show in which cases mPlane locates service degradation. These cases can be identified by a sudden change in RTT and throughput.

3.7.2 Components mapping into the demo architecture

The demo of this use case will include the following components:

- The probe: the probe that will be used is mSLAcert, it has two components, a server and a client, as published on mPlane web site. The probe performs Ping tests using ICMP, TCP and UDP tests using Iperf to generate and measure the traffic. For the probe two PCs will be needed, one will act as a server and the other one as a client. The two PCs have to be directly reachable by means of an IP connection. The probe will measure the minimum, mean and maximum of RTT, throughput and capacity between server and client. The client probe will only register to the supervisor and will start TCP and UDP service.

- The repository: the SLA will need just a network HDD that can be reachable by the PC where the reasoner will be. Alternatively, the data could be stored on the same PC as the Probe Server side.

- Reasoner: The reasoner will be hosted on a third PC, or in the same PC as the repository and the Probe server side. The role of the reasoner will be to recheck the continuous measurements and release a PDF certificate with the measurements.

- Supervisor: the SLA use case will not have a specific supervisor, the one that comes with the software delivery will be used. The supervisor will be hosted on a fourth PC, or it could be stored on the same PC as the reasoner. So, in case not many machines are available, only two machines will be needed to run the demo for the SLA use case. One machine will host the Supervisor, Reasoner and Probe Server side and the other PC will host the Probe client side.

- Client: the mPlane client has not been considered yet for the demo of this use case.

3.7.3 Vantage points mapping

The server mSLAcert probe has to be placed within the mPlane core network and it has to have a high bandwidth, at least 100Mbps, 1Gbps suggested. The client mSLAcert probe has to be placed near/at the access network. The other components could be located at the mPlane core network.

3.7.4 Demo execution flowchart

Fig. 6 shows a flow chart on the demo for this use case:

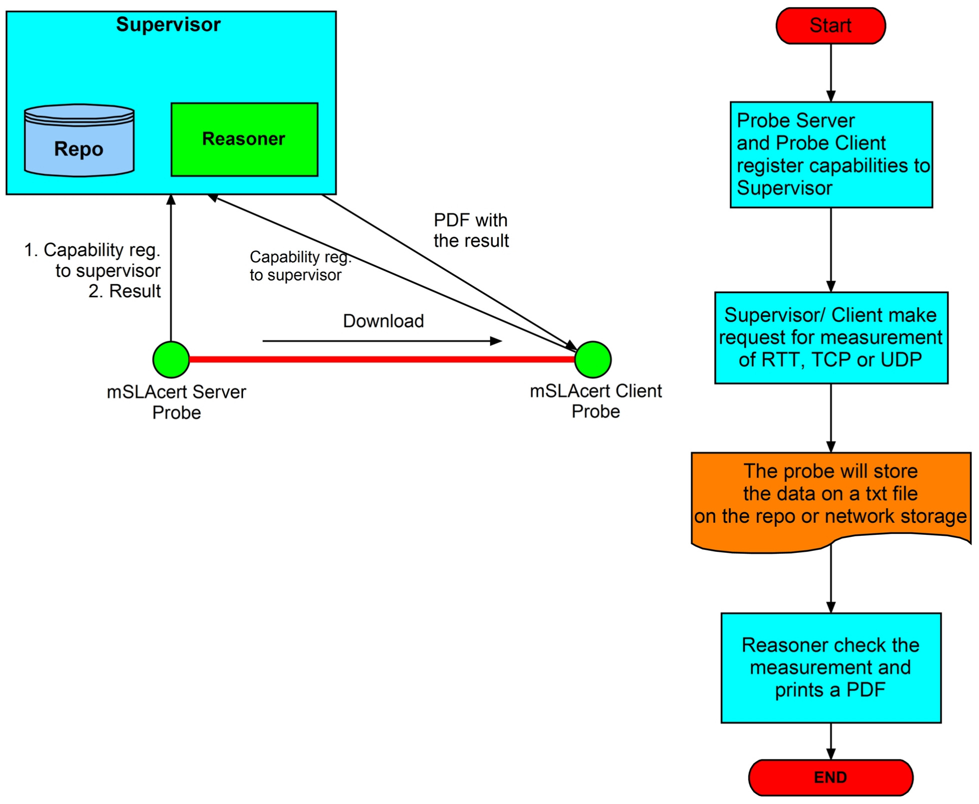
Supervisor
Repo
Reasoner
1. Capability reg. to supervisor 2. Result
Capability reg. to supervisor
PDF with the result
Download
mSLAcert Server Probe
mSLAcert Client Probe
Start
Probe Server and Probe Client register capabilities to Supervisor
Supervisor/ Client make request for measurement of RTT, TCP or UDP
The probe will store the data on a txt file on the repo or network storage
Reasoner check the measurement and prints a PDF
END

Figure 6: On the left an overview of the components and communication between them, on the right the flowchart for the demo execution.

A Hardware list

In this appendix it is presented a short list of the hardware made available in FASTWEB for the mPlane demo.

Hostname	Functionality	Model	Processor	RAM (GB)	HDD (TB)
bk501mpl	NAS	QNAP TS-EC1679U-RP	Xeon E3-1225 4 core	4	40
pm501mpl	Tstat probe 1	Desktop PC Supermicro	Xeon E3-1270 8 core	8	4
pm502mpl	Tstat probe 2	HP DL380p Gen8	Xeon E5-2543v2 6 core	16	4
as501mpl	Supervisor	HP DL360 G5	Xeon 5160 4 core	12	1,3
as502mpl	Reasoner	HP DL360 G5	Xeon 5160 4 core	50	1,3
as503mpl	Hadoop Cluster NameNode	HP DL360 G5	Xeon 5160 4 core	50	1
as504mpl	Hadoop Cluster Node	HP DL360 G5	Xeon E5420 8 core	26	1
as505mpl	Hadoop Cluster Node	HP DL360 G5	Xeon E5420 8 core	26	1
as506mpl	Hadoop Cluster Node	HP DL360 G5	Xeon E5420 8 core	26	1
as507mpl	Hadoop Cluster Node	HP DL360 G5	Xeon E5420 8 core	26	1
as508mpl	Hadoop Cluster Node	HP DL360 G5	Xeon E5420 8 core	26	1
as509mpl	Hadoop Cluster Node	HP DL360 G5	Xeon E5420 8 core	26	1
db501mpl	Data base server (dBStream)	HP DL380p Gen8	2 x Xeon E5-2620 6 core	128	24
am501mpl	active measurement	Mikro Tik Probe M-180			
am502mpl	active measurement	Mikro Tik Probe M-195			
am503mpl	active measurement	Mikro Tik Probe M-195			
am504mpl	active measurement	PC client	Intel I5	2	0,24
am505mpl	active measurement	PC client	Intel I5	2	0,24
am506mpl	active measurement	PC client	AMD Athlon	2	0,24

References

[1] P. Casas, A. D'Alconzo, M. Dusi, S. Nikitaki, M. Ahmed, S. Traverso, M. Mellia, D. Apiletti, L. Grimaudo, E. Baralis, D. Rossi, D. Joumblatt, A. Capello, M. D'Ambrosio, F. Invernizzi, M. Ullio, A. Fregosi, E. Kowallik, S. Raffaglio, A. Sannino, M. Milanesio, E. Tego, F. Matera, T. Szemethy, B. Szabo, L. Németh, Z. B. Houidi, G. Dimopoulos, I. Leontiadis, Y. Grunenberger, L. Baltrunas, M. Faath, R. Winter, and D. Papadimitriou. Design of the reasoner. (D4.2), 06/2014 2014.

[2] M. Dusi, S. Niccolini, S. Nikitaki, D. Apiletti, E. Baralis, A. Finamore, L. Grimaudo, S. Traverso, F. Matera, E. Tego, G. V, Z. B. Houidi, P. Michiardi, M. Milanesio, Y. Gong, D. Rossi, I. Leontiadis, D. G, T. Szemethy, B. A, A. Bär, P. Casas, A. D'Alconzo, and P. Fiadino. Algorithm and scheduler design and implementation. 09/2014 2014.

[3] Google. https://support.google.com/news/publisher/answer/40392.

[4] I. G. Young. The code that powers reddit.com. https://github.com/iangreenleaf/reddit.