Programme ANR VERSO

Projet VIPEER

Ingénierie du trafic vidéo en intradomaine basée sur les paradigmes du Pair à Pair

Décision n° 2009 VERSO 014 01 à 06 du 22 décembre 2009
T0 administratif = 15 Novembre 2009
T0 technique = 1er Janvier 2010

Livrable 5.5

Evaluation of the demonstrations

Auteurs:
M.K Sbai, G Madec, P Mitharwal, A Gravey , G Simon(Telecom Bretagne), J Kypreos(Envivio) , F Guillemin, S Moteau, P Philippe (OrangeLab), Y Hadjadjaoul (Inria), J Garnier (NDS)
Editeur:
M.K Sbai, G Madec, , A Gravey (Telecom Bretagne)

Décembre 2012

Telecom Bretagne; Eurecom; INRIA; France Telecom; NDS; ENVIVIO

Abstract

The main objective of the VIPEER project is to provide methods allowing a network operator to have explicit control on traffic flows related to video distribution. The work package 5 mainly consists of implementing and deploying the VIPEER architecture on an inter-partner platform. It also designs the test scenarios, demonstrations and evaluates the obtained results. This document describes the evaluation of the demonstrations. This evaluation is based on implementing, running and testing the final demonstration scenarios described in D5.3.

Contents

1 Introduction

The main objective of VIPEER is to allow NSPs control video traffic transiting in their domains. On one hand, VIPEER allow them to minimize the inter-domain traffic by deploying some streaming servers inside the operator domain. On the other hand, clients will be server by a set of servers constructing together a local CDN called dCDN. The selection of the streaming server (dCDN server) when asking for a new chunk plays a major role in optimizing the video traffic and in the quality of experience (QoE) perceived by the clients. Different software components implicated in a dCDN have been developed inside the VIPEER consortium and have been tested on a platform that consists of an interconnection of Project partner-sites.

This deliverable is the last one from the WP5. In this document, we describe the experiments and their results. The objective is to evaluate the implemented components on the established test architecture. The experiments have been conducted on inter-partner platform described in Section 2. The different software components developed for the purpose of experimentation have been deployed on different machines located at these sites. A reminder of these components and their placement on different partner machines is described in Section 3. Using this material and software testbed, we was able to run different test scenarios described in D5.3. They are summarized in Section 4. The evaluation of the tests and their results is discussed in details in Section 5.

2 Architecture of the platform

The partners of the project VIPEER decided to inter-connect some machines located at their different sites and to dedicate them for the test experiments . These machines connected together form an emulation of an operator network mainly the machines represent the intra-domain streaming servers and clients. The sites implicated in this platform are:

- TELECOM Bretagne, Brest, France
- Orange Labs, Lannion, France
- INRIA, Rennes, France

NDS Limited, Paris has rented a dedicated server at Amazon to play the role of the original external CDN. This server is located at Dublin, Ireland.

The interconnection network is the public Internet. Each of the server machines is addressed by a public IP address. The clients can be located behind NAT servers. Table 1 describes the different machines implicated in the platform.

Figure 1 locates the different machines of the dCDN and the original CDN on a map.

Table 1: Machines of the platform

Site	Machine Label	IP Address	OS	Special packages	Main role
INRIA, Rennes	Linux machine Rennes	131.*.*.34	Linux	Oracle java 7 Tomcat 6	dCDN streaming server
	Windows machine Rennes	*.*.*.* not public	Windows	Oracle java 7	streaming clients
Orange Labs, Lannion	Linux machine Lannion	193.*.*.6	Linux	Oracle java 7 Tomcat 6	dCDN streaming server
	Windows machine Lannion	*.*.*.* not public	Windows	Oracle java 7	streaming clients
TELECOM Bretagne, Brest	Linux machine Brest 1	193.*.*.203	Linux	Oracle java 7 Tomcat 6	dCDN streaming server
	Linux machine Brest 2	193.*.*.204	Linux	Oracle java 7 Tomcat 6 MySQL server	dTracker
	Windows machine Brest	*.*.*.* not public	Windows	Oracle java 7	streaming clients
Amazon, Dublin, Ireland	Linux machine Amazon	54.*.*.156	Linux	Oracle java 7 Tomcat 6	CDN streaming server

3 Actors: Software components

In this section, we summarize the software components and their deployment on different machines of the platform. These components can be divided following their roles into two sets: **Monitoring components**, **Scenario-controlling components** and **Streaming components**; and following their locations into 3 sets: **Streaming server-side components**, **Client-side components** and **Central intelligence components**.

3.1 Streaming components

Streaming software components are components that together implement the streaming functionalities.

Central intelligence components:

- **MPD constructor Servlet**
  - *Location: Linux machine Brest 2*
  - *Role:* When a client asks to stream a new video to the original CDN server, it must download a media description file called MPD. This file

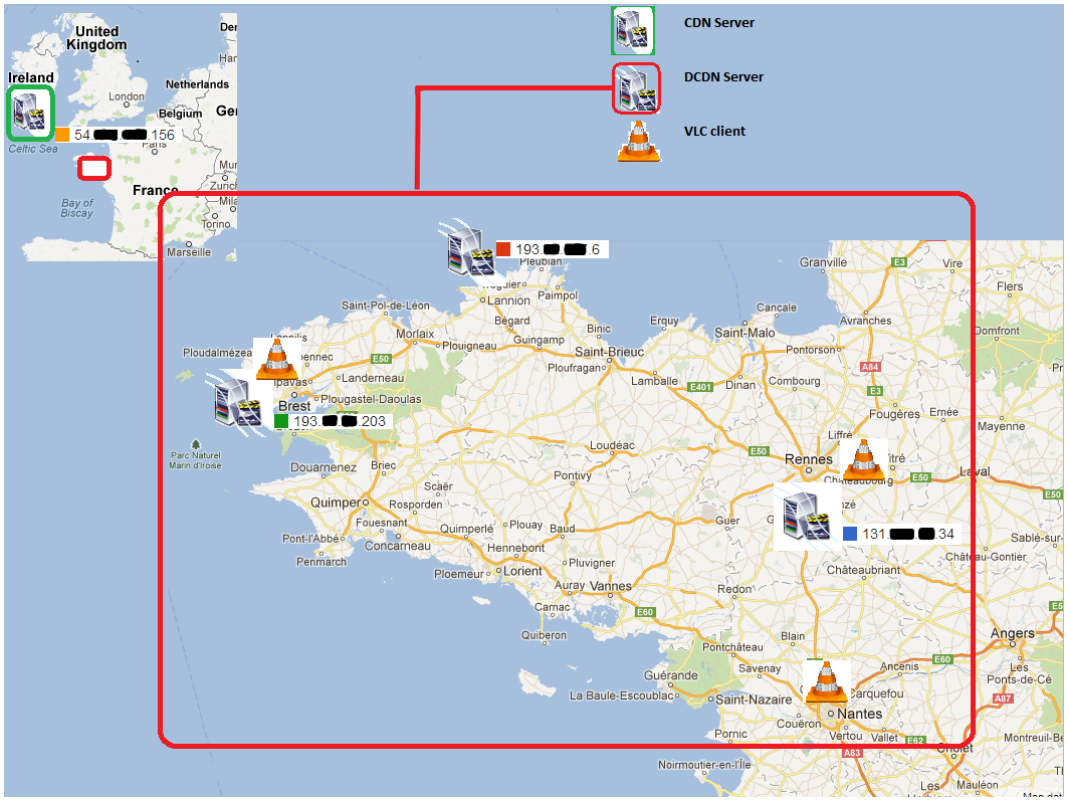

Figure 1: Testbed architecture

contains the URLs of the different chunks of the video. In the case of the presence of a dCDN, the original CDN server redirects the client to the MPD constructor Servlet which constructs an MPD specific to the client. All the URLs of the chunks, in this MPD, are links to a Redirection Servlet that will redirect it later to the best chunk server.

  - *Technology:* Java Web Servlet

- **Redirection Servlet**
  - *Location:* Linux machine Brest 2
  - *Role:* When requesting a chunk, the client contacts the Redirection Servlet which computes following a specific selection strategy the best dCDN server from where to get the chunk. The information of the current placement of the chunks and the network/server conditions are stored in the system's database.
  - *Technology:* Java Web Servlet
- **System's database**
  - *Location:* Linux machine Brest 2
  - *Role:* This database contains information about the placement of chunks, the different available servers, monitoring information, etc
  - *Technology:* MySQL server

Streaming server-side components:

- **Tomcat HTTP server**
  - *Locations:* Linux machine Brest 1, Linux machine Rennes, Linux machine Lannion, Linux machine Amazon
  - *Role:* It is a HTTP server that serves chunks of different videos organized in its local system folders.
  - *Technology:* Web server

Client-side components:

- **VLC DASH client**
  - *Locations:* Windows machine Brest , Windows machine Rennes, Windows machine Lannion
  - *Role:* The VLC DASH client is a classical adaptive http streaming client. It is supposed to adapt the bitrate of the requested chunks to the delay of the reception of chunks. However, the current version of VLC is not really compliant to the automatic regulation of the bitrate as stated in the standard.
  - *Technology:* MPEG-DASH standard to be published as ISO/IEC 23009, 2013. The VLC client version used is: VLC 2.1.0

3.2 Scenario-controlling components

Scenario-controlling software components are components that allow to run remotely streaming sessions on clients located at partner sites.

Central intelligence components:

- **Scenario Controller**
  - *Locations:* Linux machine Brest 2
  - *Role:* This is a server to whom different Session Launchers located at client machines connect. It reads the scenarios from the system's database and send "START", "STOP" scenario/session messages to the Launchers. The Scenarios are entred by a user through the system's graphic interface.
  - *Technology:* Java, TCP sockets
- **Web interface**
  - *Locations:* Linux machine Brest 2
  - *Role:* The web interface allows to introduce the scenario (streaming session) to be run and to follow the results in real time.
  - *Technology:* Java script

Streaming server-side components:

For experimental reasons, it is possible for the "Measurement controller", described later in this section, to throttle the bandwidth of any of the dCDN Servers. It sends then a message only to the concerned "Measurement Node". The latter uses the Linux kernel TC command to limit the upload capacity of its network interface. This limitation of the upload available bandwidth is considered in our platform in order to allow experimenting with overloaded servers.

Client-side components:

- **Session Launcher**
  - *Locations:* Windows machine Brest , Windows machine Rennes, Windows machine Lannion
  - *Role:* This software component acts as a client for the Scenario controller from whom it receives the URL of the MPDs of videos to stream. It can also receive an order to stop a running streaming session.
  - *Technology:* Java, TCP sockets

3.3 Monitoring components

Monitoring software components are components that allow to run remotely streaming sessions on clients located at partner sites.

Central intelligence components:

- **Measurement Controller**
  - *Locations:* Linux machine Brest 2
  - *Role:* The Measurement Controller Server is the controller unit which communicates with all the measurement nodes located in dCDN Servers. Whenever a client gets connected/disconnected, the Controller sends a message to all measurement nodes and order them to start/stop measurements of RTT(Round Trip Time) to this client.
    Measurement components at dCDN Servers are periodically sending the current upload rate to the measurement controller in order to compute the available upload bandwidth ($Availableuploadrate = maximumuploadcapacity - uploadrate$).
  - *Technology:* Java, TCP sockets

Streaming server-side components:

- **Measurement Node**
  - *Locations:* Linux machine Brest 1, Linux machine Rennes, Linux machine Lannion, Linux machine Amazon

- *Role:* The following instructions summarize the functionalities of a measurement node located at one of the dCDN Servers:
  1. Open in-out connection for the Measurement Controller Server port.
  2. Calculate Current Upload rate using "tc" command.
  3. Read messages from the Controller:
     * If the Measurement node receives a "START measurement" message for a client, this client will be added to the active clients' list. The ping command is then used for measuring RTT between the server and all the clients present in the active list. These measurements are done periodically (a period of 10sec. has been selected in our experiments).
     * If the measurement node receives a "STOP measurement" message for a client and if it is already measuring RTT for this client, it will be deleted from the list.
     * Only in the experimental testbed, if the server receives a "Start Rate Control" message, it will control the maximum upload rate using "tc" commands.
  4. Send back results of measurements periodically to the controller.
- *Technology:* Java, TCP sockets, tc, ping

Client-side components:

No monitoring components are installed on client side.

4 Scenarios of the experiments

In this section, we describe the scenarios of the experiments which we have conducted on our platform.

4.1 The video

The video stream is temporally segmented into chunks. Each of the chunks is an encoded video with AVC (MPEG4-10 "Advanced Video Coding" / H264) using a specific bitrate. The server will provide to clients different bitrate levels.

We used the following video:

- Video Name: OfForestAndMen
- Duration: 10:53
- Chunk duration: 1s
- Chunk format: MP4
- Supported bitrates: 900 Kbit/s to 3000 Kbit/s

Figure 2 represents two screenshots taken from the video at the same moment with the two different bitrates. The system can shift the client's streaming rate when there is not enough capacity on dCDN servers.

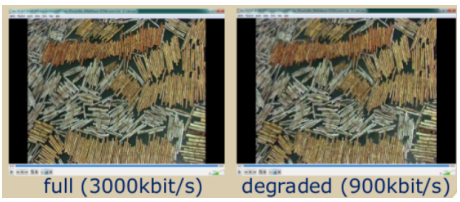

Figure 2: Video quality

4.2 Classes of clients

An operator or a content provider would like to have different classes of clients depending on their billing strategy. Two classes of clients have been defined based on much of the total capacity of a streaming server the client can see. The classes of clients is represented in Figure 3

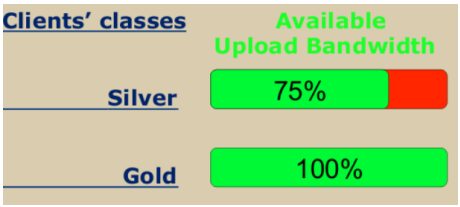

Figure 3: Classes of clients

- **Gold Client:** It sees 100% of the total capacity of streaming servers. It keeps the best rate during a longer period in case of overloaded servers.
- **Silver Client:** It sees only 75% of the total capacity of streaming servers. In case of an overloaded server, it is among the first clients to change streaming server or to downgrade the streaming rate.

4.3 Scenarios

In each of the scenarios described later, the Brest Client (Windows machine Brest) is asked by the Scenario Controller to run 4 streaming sessions of the video OfForesAndMen in parallel:

- 3 Gold streaming session with a maximum bitrate of 3000kbit/s
- 1 Silver streaming session with a maximum bitrate of 3000kbit/s

The maximum available upload bandwidths of the dCDN servers and the original CDN server are kept to their maximum at the beginning of each experiment. After

some seconds, they are gradually throttled down to reach 5 Mbit/s. In our tests, this occurs starting from the 30sec. of streaming.

Two main scenarios have been defined depending on whether we use or not the dCDN servers:

Scenario 1: Classical DASH algorithm

In this scenario, the dCDN servers are not used to stream chunks. The clients retrieve all the chunks from the original CDN server located in Ireland. The objective of this scenario is to evaluate the classical DASH algorithm. Mainly, one can see the impact of reducing the available upload capacity of the original CDN server on the bitrates of the 4 streaming sessions (3 Gold and 1 Silver). One can then conclude on the Quality of Experience perceived by the users. In this scenario, it is evident that 100% of the streaming traffic go through inter-domain links.

Scenario 2: Network-friendly DASH algorithm

In this scenario, the dCDN servers are used to stream chunks. The implemented server selection strategy is very simple. It can be summarized in three steps. If the condition in one of the steps is fulfilled, this means that a server has been selected and the algorithm of selection does not move to the next step. Otherwise, it continues with the next step:

1. If there is some dCDN servers that can stream at full rate to the client (i.e available bandwidth > video full rate), select the nearest server. The metric used for this selection is the RTT (round-trip time)
2. If there is some dCDN servers that can stream at degraded rate to the client (available bandwidth > degraded), select the nearest server. The metric used for this selection is the RTT (round-trip time).
3. Redirect the client to the original CDN server.

Figure 4 plots the server selection strategy at the dTracker level.

The objective here is to evaluate the impact of using the dCDN servers on the inter-domain traffic and on the quality of service received by clients.

5 Results and evaluation

In this section, we describe the results of the experiments conducted to test the two scenarios described earlier. Then, we evaluate this results from the VIPEER project perspectives.

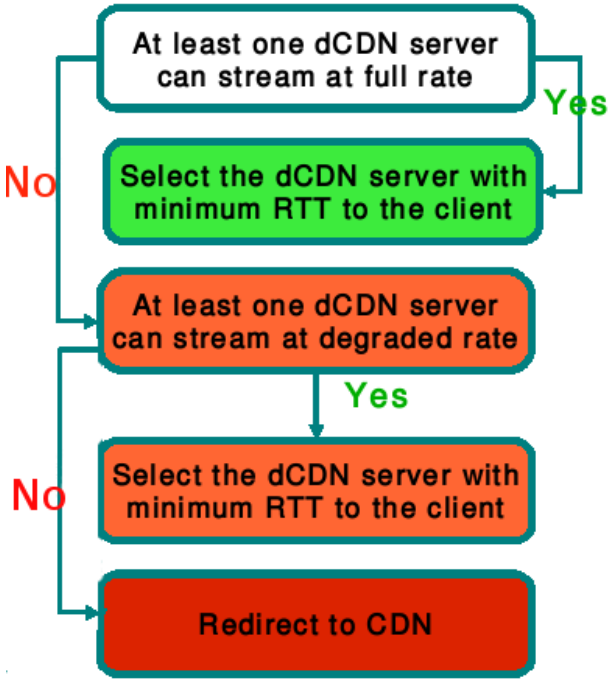

Figure 4: dCDN Server Selection strategy

5.1 Results of experiments

Results of experiments in case of Scenario 1

In this scenario, we run three streaming sessions (3 Golds and 1 Silver) in parallel. The chunks are exclusively retrieved from the original server. Figure 5 plots the streaming bitrate of the four sessions as a function of session time. It shows that at the beginning of each session the rate is at the maximum 3000 kbit/s then after 30sec. when the bandwidth throttling occurs the Silver client is the first to diminish its bitrates to 900 kbit/s. In fact, it sees only 75% of the available upload rate on the original CDN server. After some seconds, the Gold session begin also to diminish their bitrates as the global available upload rates is not enough to run 3 sessions in parralel. The available upload rate on the CDN server is shown in Figure 6. It shows that at second 50 the available upload rate is 5000 Mbit/s.

In this scenario, the streaming traffic is 100% inter-domain as the dCDN servers are not activated. This can be seen in Figure 6 where the available upload rates of the three dCDN rates do not varie and stay at their maximum values.

Results of experiments in case of Scenario 2

In this scenario, we run three streaming sessions (3 Golds and 1 Silver) in parallel. The chunks can be retrieved either from the dCDN servers or the original CDN server following the selection strategy explained earlier in this document. Figure 7 plots the streaming bitrate of the four sessions as a function of session time. It shows that at the beginning of each session the rate is at the maximum 3000

Figure 5: Rate of the Chunks during streaming session for different classes of clients

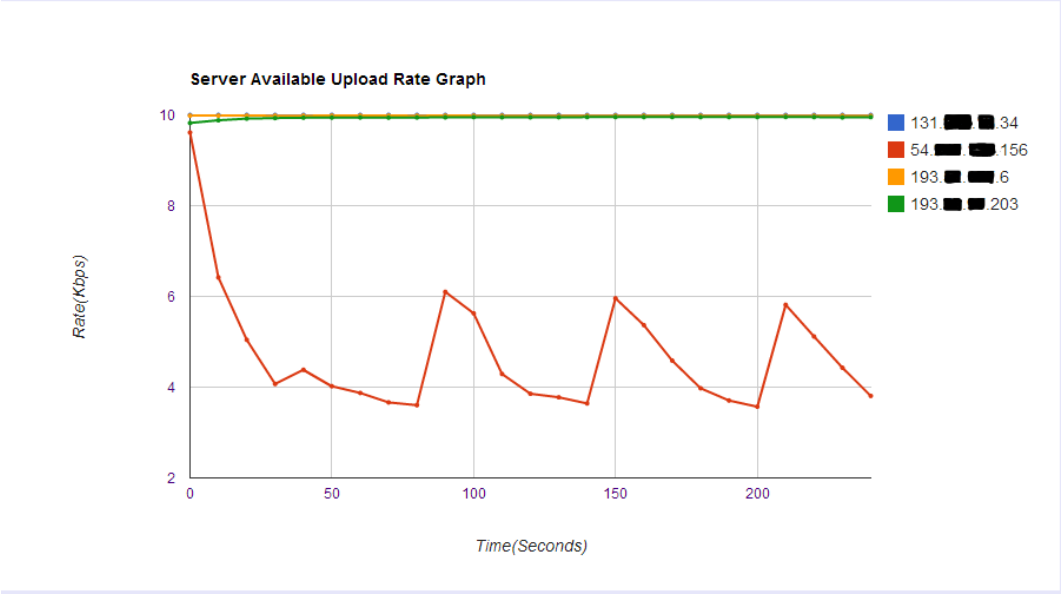

Figure 6: Available Upload bandwidth of different servers

kbit/s then after 30sec. when the bandwidth throttling occurs (see figure 8) all the clients even the Silver one is still streaming at the highest rate. Hence, our optimized DASH algorithm allows to serve clients with a better quality of service. The following results focus on the available upload rate and RTT metrics measured by the monitoring components during the same experiments.

Figure 9 plots the Round-Trip Time (RTT) metric measured between two clients and different streaming servers (the CDN server (54.*.*.156) and the dCDN servers

located at partner sites) during a streaming experiment. The first client is located at INRIA, Rennes and the second one at Telecom Bretagne, Brest. The figure shows that even if a server can be the nearest to a client at the beginning of a streaming session, its RTT can become greater when the load of streaming requests increases. That is why, it is worthy for the system to select another farther dCDN server for the client. For instance, at the beginning of a session, the nearest server for the client located at Brest was, as shown in the figure, the brest dCDN server. When the Brest server has become overloaded, the RTT between the Brest client and the Brest server has become greater than the RTT between it and the Lannion dCDN server. One can imagine that at this moment the Brest client will be streaming chunks from the Lannion dCDN server.

The second monitored metric is the available bandwidth at servers. It is shown in Figure 8 which shows mainly that, unlike the original CDN server, dCDNs servers are used to stream chunks . The load of the streaming task is almost equilibrated between the three dCDN servers. Hence, there is almost no inter-domain traffic which one of the main goals of VIPEER.

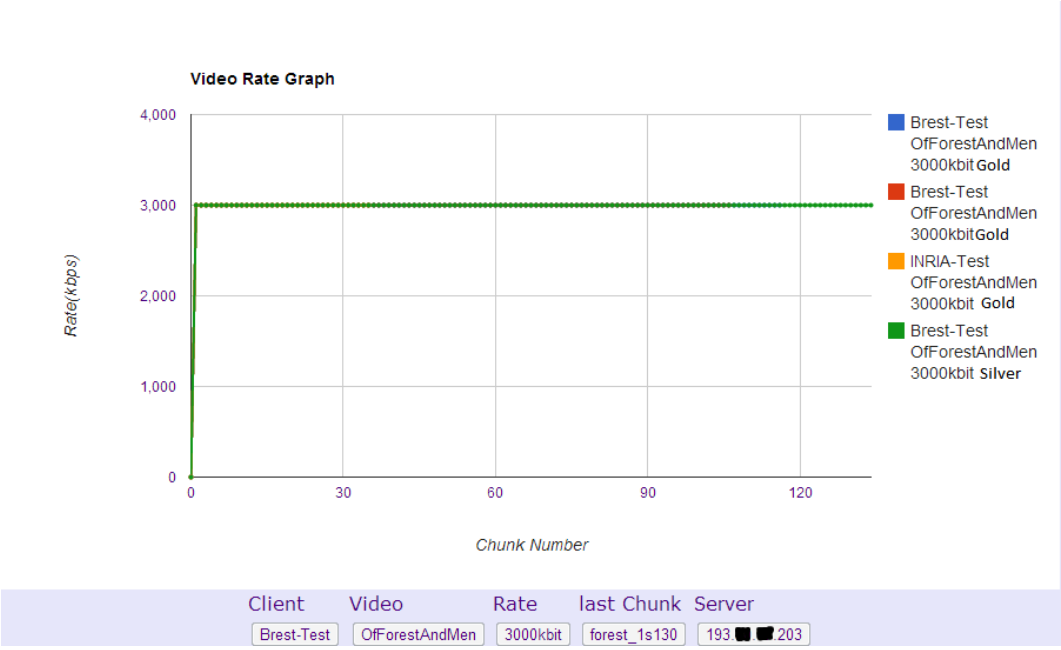

Figure 7: Rate of the Chunks during streaming session for different classes of clients

5.2 Evaluation

The criteria of the evaluation are:

- Feasibility of the scenario:
  - Communication between partner sites
  - Measurement of network conditions and QoS

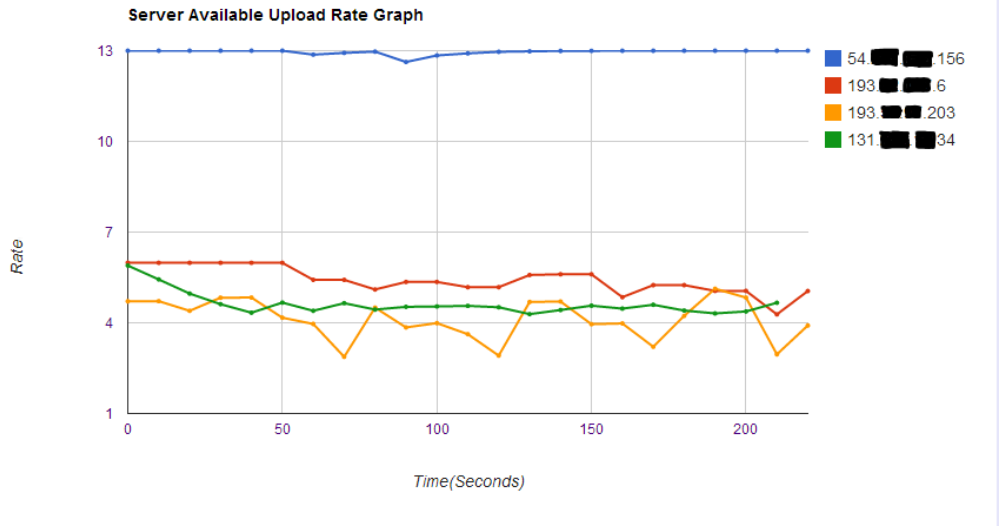

Figure 8: Available Upload bandwidth of different servers

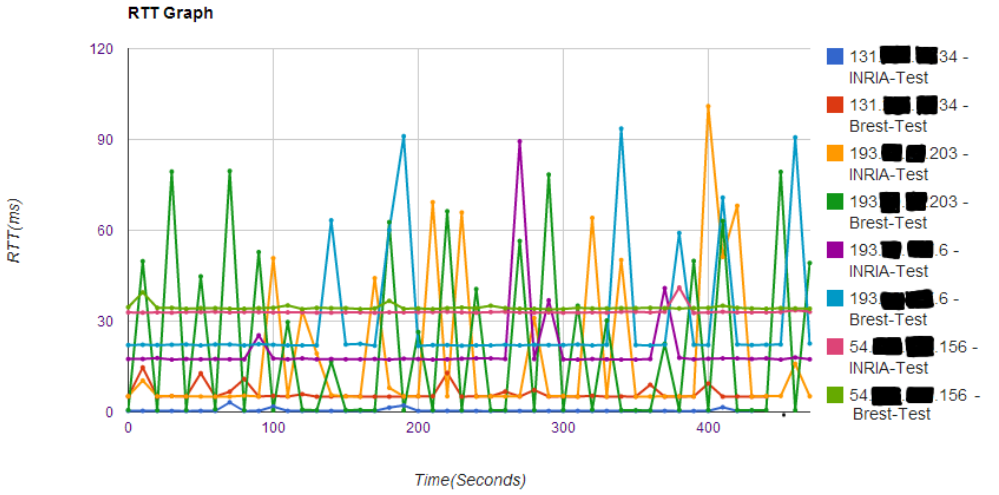

Figure 9: RTT between clients and servers

- Performance and Efficiency:
  - Efficiency of the video streaming from client point of view
  - Efficiency from network point of view

Evaluation in case of Scenario 1

- **Feasibility of the scenario:**

  - **Communication between partner sites:** OK with public IP addresses. Clients located at Brest can connect easily to the Amazon CDN server (54.*.*.156)
  - **Measurement of network conditions and QoS:** OK. Our measurement tools are running both on client side and server side. (Figure 6 shows a tracking of the available upload bandwidth on the original server)

- **Performance and Efficiency:**

  - **Efficiency of the video streaming from client point of view:** As the classical algorithm is run, while there is enough available upload bandwidth on the path and the server capacity is not reached, the client can keep streaming at the highest rate (3000 kbit/s). When the server upload capacity is delimited the clients are obliged to diminish the quality it streams to 900 kbit/s. Figure 5 shows that silver clients are the first to change the rate as they see only 75% of the total capacity of the server, the gold clients are constantly changing from best to medium rate. The tests are OK but they show that our optimization using dCDN servers is worthy.
  - **Efficiency from network point of view:** We see here that for all chunks and for all clients the traffic (See Figure 6) the traffic comes from the CDN server that means that the classical algorithm does not take into consideration any NSP cost consideration. The traffic is 100% originating from a server located at another carrier. Thus, a very important inter-carrier traffic.

Evaluation in case of Scenario 2

- **Feasibility of the scenario:**

  - **Communication between partner sites:** OK with public IP addresses
    * Clients located at Brest can connect easily to the Amazon CDN server (54.*.*.156)
    * Clients located at Brest can connect easily to servers located at TB Brest Orange Labs Lannion and INRIA Rennes.
  - **Measurement of network conditions and QoS:** OK. Our measurement tools are running both on client side and server side. Figure 8 and 9 show respectively a tracking of the available upload bandwidth on the original server and dCDN servers; and the delays (RTT) between clients and different servers

- **Performance and Efficiency:**

  - **Efficiency of the video streaming from client point of view:** As dCDN servers are activated the client will be served from the nearest available server. The available bandwidth between the selected server and the client is the best at any moment of the streaming session. Figure

7 shows that for all the streaming servers (gold and silver) keep streaming at the highest rate even if we limit the capacity of the original server (or any other dCDN server); it always finds the best location to stream the current chunk.

– **Efficiency from network point of view:** As figure 8 show the dCDN servers are solicited more than the original server there is a gain of at least 95% on inter-carrier traffic. And also the video traffic inside the operator network is controlled by it.

6 Conclusions

In this deliverable, we presented the test scenarios and their evaluation. Our experiments allowed us to validate that we have fulfilled many of the initial goals of the VIPEER project. For instance, we show that deploying a local CDN (dCDN) in the operator network allows to economize 95% of transit traffic and to have better quality of service observed by streaming clients. Future work in experimentation and validation can be adding new user-level metrics such as QoE opinion main scores both for the decision and for the monitoring phases.