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 2.3

Advances on monitoring primitives and integration in the VIPEER prototype

Auteurs:
T.Groléat, M.K.Sbai, S.Vaton (Télécom Bretagne),
Y.Hadjadj-Aoul, K. Singh (INRIA),
S.Moteau (France Télécom)
Editeur:
S.Vaton (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 2 mainly consists of building a global measurement infrastructure in order to inform the network and QoS aware dCDN with useful information about the network state and about the end user activity.The ultimate goal is to be able to optimize the distribution strategy of the dCDN so as to minimize the impact of the dCDN on the QoS perceived by the set-top-box owners and to maximize the QoS experienced by the dCDN's users.Thus, different tools and algorithms are developed within this work package. This last deliverable presents first the technical specifications of the last versions of traffic classifier and of the QoE evaluation module and secondly depicts the integration of WP2 work in the final demonstrator.

Keywords: network monitoring, QoE, traffic classification, demonstrator, network metrics

Contents

Contents 4

1 Introduction 7

2 QoE model for varying quantization values over different chunks 9
2.1 Introduction . . . 9
2.2 Video database generation and subjective testing . . . 9
2.3 QoE Model . . . 11
2.4 Conclusion . . . 13

3 Advances in traffic classification 15
3.1 Introduction . . . 15
3.2 SVM based classification . . . 15
3.2.1 Principle of SVM based classification . . . 15
3.2.2 An algorithmic view of SVM . . . 16
3.2.3 Performances of SVM based classification . . . 16
3.3 Need for hardware acceleration . . . 18
3.3.1 Limits of a software implementation of SVM . . . 19
3.3.2 Requirements for the hardware accelerated classifier . . . 20
3.4 Design of the hardware accelerated classifier . . . 21
3.4.1 Fixed point data representation . . . 21
3.4.2 Hardware architecture . . . 22
3.4.3 Synthesis results . . . 23
3.5 Validation on generated traffic and on a campus network . . . 24
3.5.1 Design of a hardware accelerated traffic generator . . . 24
3.5.2 Validation on a campus network . . . 25
3.6 Conclusion . . . 27

4 Integration of Monitoring Metrics in the VIPEER Demonstrator 29
4.1 Introduction . . . 29
4.2 Metrics . . . 30
4.2.1 Round-Trip Time (RTT) . . . 30
4.2.2 Available upload Bandwidth . . . 31
4.3 Measurement architecture . . . 31
4.3.1 Measurement Controller . . . 31
4.3.2 Measurement Node . . . 32
4.4 Example of measurement results . . . 33

4.5 Conclusions and Future work 35

5 Conclusion 37

Bibliography 39

1 Introduction

This deliverable is the last one from the WP2. It provides the technical specifications for the different tools of the global measurement infrastructure and provides results of these tools regarding real data and/or experiments. As additional metrics were needed to take decision regarding to the chunk placement and duplication (works of WP4), this deliverable deals too with monitoring metrics and their integration in the VIPEER demonstrator.

In the VIPEER context, the measurement infrastructure tools have to inform the central intelligence of the dCDN about network state and end user activity in real time. This knowledge is useful to determine both the optimal storing strategy and the best reaction to quality degradation of services (selection of the appropriate replica of chunk with a suitable quality level).The ultimate goal is to be able to optimize the distribution strategy of the dCDN so as to first minimize the impact of the dCDN on the QoS perceived by the set-top-box owners and to maximize the QoS experienced by the dCDN's users and second to deliver content with the best QoE possible regarding to network troubles.

Measurement of network performance parameters is provided by state of the art techniques such as packet pair and other techniques that can be classified as either active or passive. The tools used for traffic classification and QoE estimation are based on supervised learning techniques. Traffic classification can be done using Deep Packet Inspection (DPI), but it is extremely demanding on high bandwidth links and cannot be used if the applications cipher their traffic. Thus, a classifier based on Support vector machine (SVM) is used that employs statistical analysis of some traffic descriptors such as the length of the first data packets in order to classify flows. In the chapter dealing with this topic, new results are presented: hardware acceleration of SVM based traffic classification. For QoE estimation a tool called PSQA (pseudo subjective quality analysis) based on random neural networks (RNN) is used. In the chapter dealing with this topic, implementation details and evaluations are given. A new topic in WP2 is the integration of monitoring in the demonstrator. This concerns the monitoring of QoS metrics (Round-Trip Time RTT and available upload bandwidth), design of the measurement architecture and real experiments.
More details about these tools are provided in the following chapters. Chapter 2 deals with the QoE monitoring, chapter 3 focuses on the traffic classification tool and chapter 4 decribes the integration of monitoring in the demonstrator.

2 QoE model for varying quantization values over different chunks

2.1 Introduction

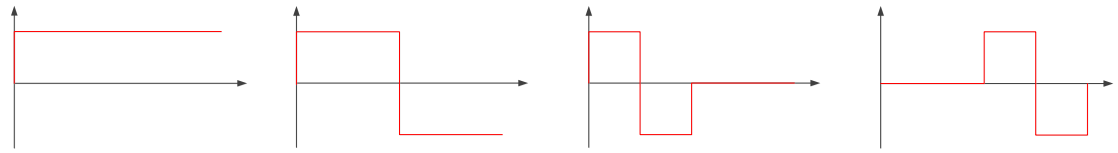
The recent adoption of adaptive streaming over HTTP, by many streaming platforms, is a direct consequence of the robustness of such protocols to cope with the vagaries of the Internet. Indeed, by smoothly degrading the quality of the received video, at the terminal level, one can completely avoid playout interruptions, which may allow increasing the user's perceived quality. In fact, as established in our previous paper [12], users are more sensitive to video playout interruptions as compared to video quality degradation resulting from increasing the quantization factor QP.

In the previous work [12], we didn't consider the degradation resulting from having chunks with different qualities (i.e. variable QP) as we only consider average values of the QP over the measurement window. In this work, instead of assuming constant values for QP, we consider a window of chunks-based QP variation. This will clearly allow improving the accuracy of the previously proposed QoE evaluator for adaptive HTTP video streaming using H.264/AVC. In fact, we found that even if the average value of QP over all chunks in the window remains the same the perceived video quality can vary due to QP variations over different chunk. We model this behavior and capture the impact of QP variations over different chunks.

The remainder of this chapter is organized as follows. Section 2.2 introduces the new generated video database which is used for subjective testing. Section 2.3 presents the proposed QoE model, which considers variable QP factor from chunk to chunk. It also presents and discusses the obtained results. Finally, section 2.4 is a conclusion.

2.2 Video database generation and subjective testing

In order to evaluate the performance of the proposed model, we have built a new video database, of 150 videos, in which a variable QP is considered. To generate different patterns of QP variations we have used a technique similar to HAAR wavelets [7], which are a group of square-shaped waves with a magnitude of ± 1. These wavelets are able to compactly represent the considered QP variations as we can see from figure 2.1. In fact, the wavelets, from the left to the right, represent respectively constant QP values, decreased QP values, variable QP in the first two chunks and variable QP at the two last chunks. Using these basis functions, all



(a) ψ_0 (b) ψ_1 (c) ψ_2 (d) ψ_3

Figure 2.1: Set of Haar basis functions

the QP patterns variations can be modeled. Note that we assume a measurement window of 8 seconds with chunks containing video of a 2s duration. With this we are able to consider chunk sizes of 2s, 4s and 8s. Note that we do not consider video of more than 8s because 8s corresponds to the recommended duration of videos for subjective testing by the ITU [1]. It is not recommended to use videos of very long durations as users tend to forget the past video quality and tend to only focus on recent video segments. For videos of more than 8s our tool will break into 8 second segments and provide a quality score for each video segment.

We use HAAR wavelets so that we are able to analyze the impact of QP variation in frequency as well as time. For example if the perceived quality is found to be very sensitive to the coefficient of the 2nd waveform (obtained after HAAR transform of data) in Figure 2.1 then we can say that quality is sensitive to QP changes over a frequency of 4s. Moreover, the impact of QP over time can also be analyzed by looking at the coefficient values when they are positive or negative.

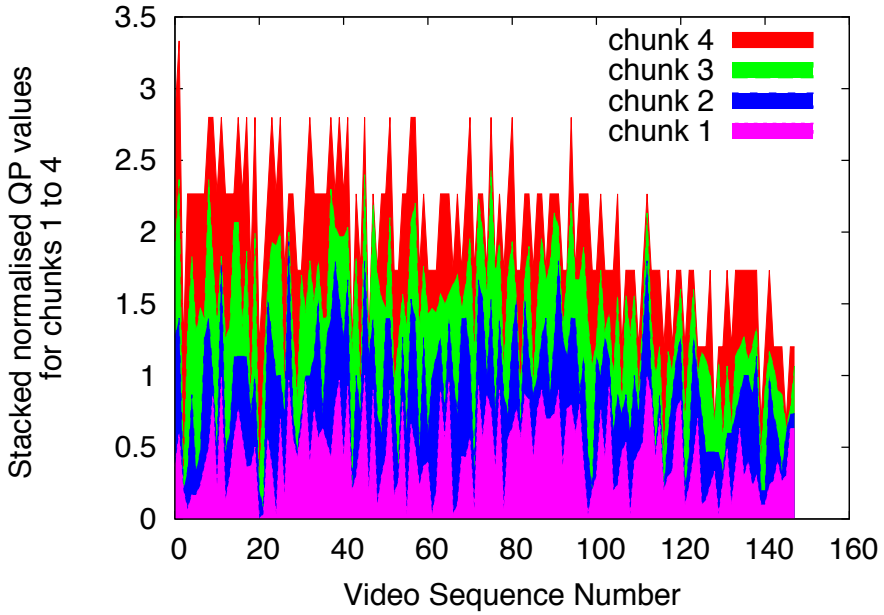


Figure 2.2: Stacked normalised QP values

In order to generate videos, we have first generated all the possible values of HAAR coefficients, let us denote them as h_1, h_2, h_3, h_4. We have let them vary as

follows: h_1 from 38 to 100 with a delta of 8 (38 here corresponds to average QP of 19 and 100 as 50 and so on), h_2, h_3, h_4 from -24 to 25 with a delta of 4. Then we have first eliminated the cases where the QP of a chunk was below 19 or above 100. This is because humans don't perceive the difference in quality if QP is further decreased from 19. The cut off value of 100 is taken because the equivalent value in terms of actual average QP = 50 is almost the max value of QP for H.264. After that we did random sampling to reduce the number of videos to an acceptable number = 150 as it is difficult to do subjective testing for too many videos. Random sampling is useful as it can uniformly sample data over different dimensions, however one drawback is that some points can be missing, creating some "holes" in the 4D space (considering 4 parameters h_1, h_2, h_3 and h_4) that may affect the prediction accuracy of the model in those regions. Thus, we have filled these holes by manually adding some videos to the sampled database after having visualized the data with different combinations of input parameters.

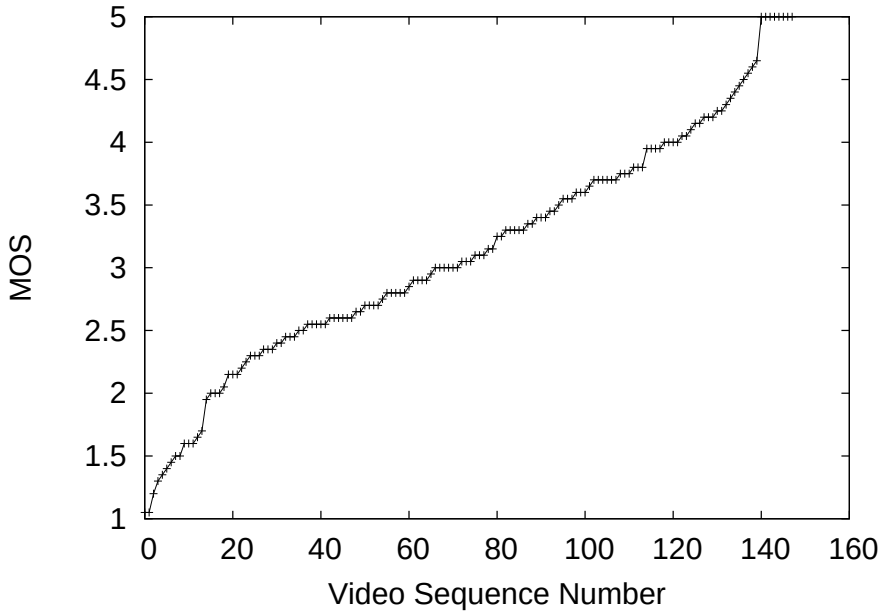


Figure 2.3: QP of the videos within the database

The QP values of the obtained videos are shown in figure 2.2, in which the values are normalized and stacked over each other. After that we have performed subjective testing for the videos such that a typical MOS scale varying from 1 for very bad, to 5 for excellent, was used. The resulting videos have then been shown to a panel of users using the testing methodology described in [1]. The obtained MOS scores are shown in figure 2.3.

2.3 QoE Model

As the humans perception is more sensitive to higher distortions, in order to model perceived quality, instead of taking the average of QP values of the different chunks,

we use a pooling mechanism such as:

$$QP_{pooled} = \sqrt[n]{\frac{1}{k}\sum_{k} {QP_k}^n} \tag{2.1}$$

where k is the chunk index and n is the exponent the defined generalized mean.

This QP_{pooled} value can be generalized, by giving variable weights to the different chunks (i.e. 4 chunks) in a way to reflect more accurately the impact of the chunk index on the perceived quality. Thus, a generalized QP_{pooled} value is used in the following QoE model:

$$mos = 1 + \frac{4}{1 + exp(b\sqrt[10]{((qp_4^{n_4} + qp_3^{n_3} + qp_2^{n_2} + qp_1^{n_1})/4)} - bc)}, \tag{2.2}$$

where the values of the above parameters are determined using non linear regression analysis: $b = 0.21, c = 52.38, n_4 = 10.84, n_3 = 10.695, n_2 = 10.615, n_1 = 10.538$. Note that qp_4 has slightly more impact on MOS than other qp values this is because humans tend to forget distortions that came some time ago.

Figure 2.4 depicts the scatter plot with estimated MOS versus the real MOS obtained from subjective tests. The scatter plot clearly shows a good accuracy of the estimation. This is also reflected by the overall Root Mean Square Error (RMSE) of 0.43 for all data on the MOS scale going from 1.0 to 5.0. The RMSE is less than that of the human test panel and thus it is satisfactory.

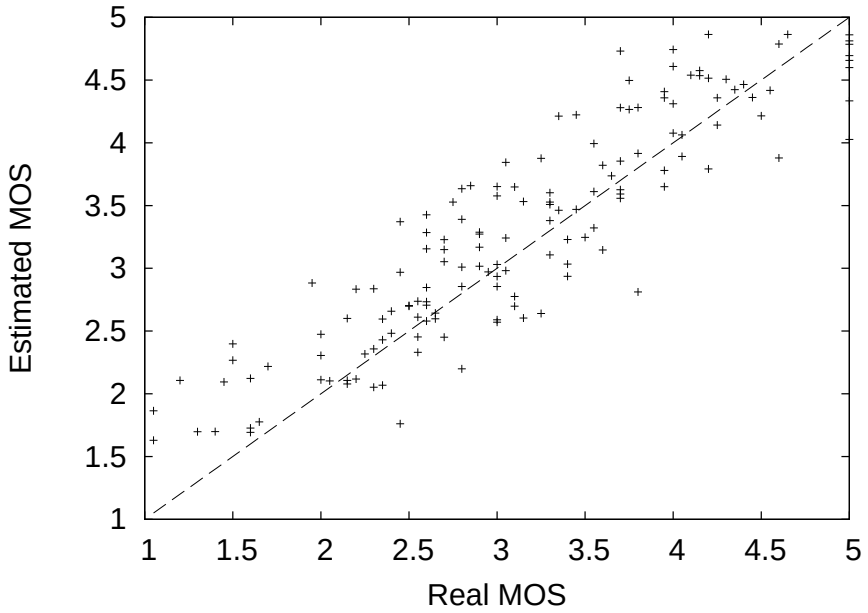


Figure 2.4: scatter plot

2.4 Conclusion

In this chapter, we have presented a solution enhancing the estimation of QoE measurement of Dash-based video streaming techniques. The proposed solution consists in taking into account the variation of the video quality over different chunks. Such variation, which was neglected in our previous work [12], is typical of the behavior of DASH-based streaming protocols. Whilst the main motivation behind this work consists in measuring the QoE of DASH-like video streaming, the proposed solution can also be used to predict the impact of a quality degradation on the perceived quality. Thus, it can be used to assist the terminal to select the best quality based on such results.

3 Advances in traffic classification

3.1 Introduction

In this chapter we are going to summarize the main developments performed and results obtained in the framework of our researches on traffic classification. Part of this chapter is based on an article [13] that we have presented during the 3rd International Workshop on Traffic Analysis and Classification (TRAC 2012), Limassol, Greece, august 2012.

Traffic classification is the task of associating network traffic with the generating application or category of applications. This is a challenging task for several reasons: (i) the limitations of traditional traffic classification methods based on port numbers or on Deep Packet Inspection (DPI), and (ii) the huge amount of traffic that operators have to analyze on the fly.

Although many classification methods such as Support Vector Machines (SVM) have demonstrated their accuracy, not enough attention has been paid to the practical implementation of lightweight classifiers. For that reason we have designed a real-time SVM classifier at many Gbps on a FPGA board to allow online detection of categories of applications.

3.2 SVM based classification

3.2.1 Principle of SVM based classification

SVM [5] separates flows in a virtual space by hyperlanes. Flows are described by simple packet level features, in our case the size of the first three data packets in the flow. As any supervised method SVM consists of two phases: *a training phase* during which the algorithm computes the classification model from a learning trace labelled with categories of applications, and *a detection phase* during which the algorithm decides of the category of application of new flows.

SVM transforms a non linear classification problem into a linear one, using a so called "kernel trick". Given a set of sample points in a multi-dimensional space one would like to separate them by hyperplanes, thus defining different classes. It is often impossible to separate sample points of different classes by hyperplanes and the separating surface is extremely difficult to compute. The idea of SVM is to map, by means of the kernel function, training points to a transformed space where it is possible to find separating hyperplanes. The output of the training phase is made up of the parameters of the kernel and a set of support vectors x_i that define

the separating hyperplane. During the detection phase SVM simply classifies new points according to the subspace they belong to.

Let us assume that we have a set of training points $x_i \in \mathbb{R}^n, i = 1, \ldots, l$ in two classes and a set of indicator values $y_i \in \{-1, +1\}$ such that $y_i = +1$ if x_i belongs to class 1 and $y_i = -1$ if x_i belongs to class 2. Let us also assume that we have selected a function ϕ such that $\phi(x_i)$ maps training point x_i into a higher dimensional space.

The training phase searches for an hyperplane that separates points $\phi(x_i)$ belonging to classes 1 and 2. The criterion is to maximize the distance of misclassified points to the separating hyperplane. The direction of the separating hyperplane is defined by a vector $w = \sum_{i=1}^{l} y_i \alpha_i \phi(x_i)$ where only a few of coefficients α_i are not null. Non-null coefficients define the so-called support vectors which characterize the separating hyperplane. The equation of the separating hyperplane is given by $w^T \phi(x) + b = 0$ that is $\sum_{i=1}^{l} y_i \alpha_i K(x_i, x) + b = 0$ where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the so called "kernel" function. A popular choice for the kernel is the Radial Basis Function (RBF) kernel which often gives good results: $K(x_i, x_j) = \exp(-\gamma \parallel x_i - x_j \parallel^2)$.

In the detection phase, any new point x is classified according to the following decision function:

$$\text{sign}(w^T \phi(x) + b) = \text{sign}(\sum_{i=1}^{l} y_i \alpha_i K(x_i, x) + b) \tag{3.1}$$

x is classified into class 1 if $w^T \phi(x) + b$ is positive and into class 2 if $w^T \phi(x) + b$ is negative.

From this simple two-class SVM problem, one can easily deal with multi-class SVM classification problems. A usual approach is the so called "one versus one" (1 vs 1) approach. In this approach $\frac{n(n-1)}{2}$ two-class SVM problem are considered, one for each pair of classes. A training phase is performed for each two-class problem thus producing $\frac{n(n-1)}{2}$ separating hyperplanes. Each new point is then classified according to each of those two-class classification problems. The final decision is taken on the basis of a majority vote, that is to say that the new point is allocated to the class which has obtained the highest number of votes.

3.2.2 An algorithmic view of SVM

The classification part of the SVM algorithm takes a vector as input and returns the class of that vector as an output. It works with few steps, repeated for each support vector. Algorithm 1 describes these steps. It is the multi-class implementation of the decision making procedure described in the previous section. This pseudo-code has been written in order to enlight the possibilities to parallelize the algorithm.

The support vectors x_i and the y, α and b values are part of the SVM model. Compared to the notations used previously, index d is added to identify the binary decision problem considered for the model values.

3.2.3 Performances of SVM based classification

In order to assess the accuracy of the SVM-based classifier we have first of all performed a validation using the libSVM library [3] over three different datasets

Algorithm 1 SVM classification algorithm

$x \leftarrow$ the vector to classify
for all support vector x_i **do** {Main loop}
  $c_i \leftarrow$ the class of x_i
  $k_i \leftarrow K(x_i, x)$
  for all class $c_j \neq c_i$ **do** {Sum loop}
    $d \leftarrow$ index of the decision between c_i and c_j
    $S_d \leftarrow S_d + y_{d,i} \times \alpha_{d,i} \times k_i$
  end for
end for
for all decision d between c_i and c_j **do** {Comparison loop}
  if $S_d - b_d > 0$ **then**
    Votes $V_i \leftarrow V_i + 1$
  else
    Votes $V_j \leftarrow V_j + 1$
  end if
end for
Select class $c_n \leftarrow$ class with the highest votes V_n

	Network	Bytes	Flows	Classified Flows	Capture mean rate (kb/s)
Ericsson	LAN Ericsson Lab.	1 755 816 952	39 056	12 858	315.18
Brescia	Campus trace Univ. Brescia	746 850 665	153 237	76 182	1 042.9
FT	DSL Link France Telecom	1 041 481 214	1 065 836	428 794	3 383.1

Table 3.1: Traffic traces and their properties

with groundtruth. The groundtruth identifies the application that has generated the traffic flow and has been obtained either by Deep Packet Inspection (DPI) with for example Linux L7-filter [4] or by using a tool such as GT [9].

The characteristics of the three traffic traces used as benchmarks are listed in Table 3.1. Those three traces correspond to three very different scenarios: campus network, laboratory environment and residential access network. As a consequence the composition of traffic is significantly different from one trace to the other.

1. The FT (France Telecom) dataset has been provided by France Telecom under the terms of a Non Disclosure Agreement. Traffic has been dumped on one geographical zone of an ADSL France Telecom access network and groundtruth has been established by DPI.

2. The Ericsson dataset corresponds to some trafic that has been generated in a laboratory environment of Ericsson research.

3. The Brescia dataset is a public dataset [9] that corresponds to some traffic captured on a campus network. The groundtruth has been obtained with the GT tool.

The definition of classes mainly depends on the filters that have been defined for DPI. In order to enable a comparison between traces we have merged applications into different categories that are listed in Table 3.2.

Class label	Class name
1	Web
2	P2P download
3	Direct download
4	Streaming
5	Game
6	Mail
7	Instant messaging
8	Distant control

Table 3.2: Traffic classes

The traffic classification accuracy, that is to say the overall percentage of flows which are correctly classified is 94.43 % for the FT trace, 98.53 % for Ericsson and 97.41 % for Brescia.

A global accuracy figure is usually not considered as sufficient to demonstrate the performance of a classifier. Some classes could be frequently misclassified with not much impact on the global figure if only few flows correspond to those classes. A usual representation of results is given by the confusion matrix. Here we provide in Figure 3.1 the accuracy per category of applications, that is to say the percentage of flows of each category of applications that has been accurately classified.

As one can see from this figure, the accuracy of the SVM algorithm differs from one category of applications to another and from one trace to another. The proportion of a category of applications in a trace impacts the ability of the SVM algorithm to detect it. For example, as class 1 (Web) is present with a good proportion in all three traces, the accuracy of the detection is high. However, as class 4 (Streaming), is almost absent in the three traces it has the worst classification accuracy. Another reason for the low classification rate of Streaming traffic might be that the size of the first packets is not an accurate descriptor for this traffic.

3.3 Need for hardware acceleration

In what follows the implementation of on-line SVM traffic classification is studied. In our scenario, only the detection phase of SVM is made on-line. The learning phase is made off-line periodically with a groundtruth generated by tools such as GT. We want to support data rates going up to tens of Gb/sec as equipments such as NetFPGA 10G [10] or COMBOv2 [2] are available to test algorithms at this speed.

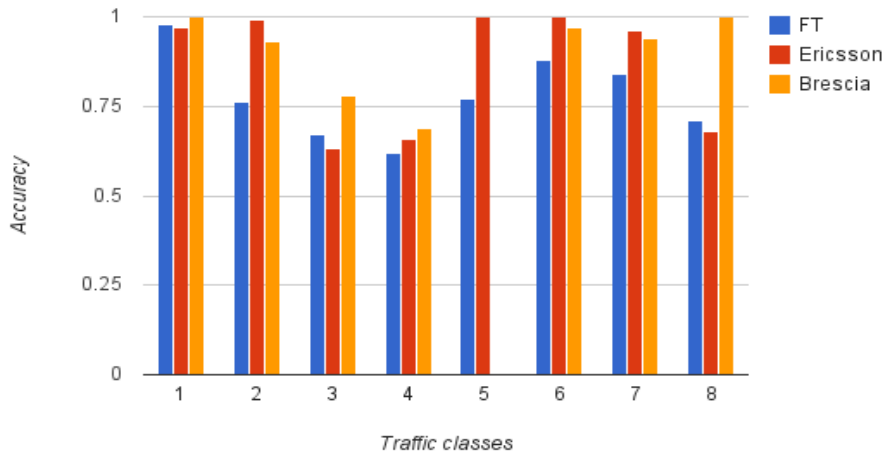


Figure 3.1: Accuracy per traffic class

The goal for on-line traffic classification is to handle all flows on a saturated 10 Gb/s link. Two main functions will be required to achieve this goal:

- The flow reconstruction reads each packet, identifies to which flow it belongs, and stores the packet lengths required for classification. The processing speed depends on the number of packets per second in the traffic.

- The SVM classification runs the SVM algorithm once for each received flow. The processing speed depends on the number of flows per second in the traffic.

3.3.1 Limits of a software implementation of SVM

We have first developed a software version of the classifier that is fed by a trace, to assess the possible performance in software. The classifier is made up of 3 main processes: (i) read the trace, (ii) rebuild flows from the stream of packets (iii) classify flows. For flow reconstruction, an algorithm proposed for a Netflow hardware implementation [14] is used. It has the advantage of requiring a constant time per packet and a bounded memory, which fits well with a hardware implementation. For the SVM algorithm, the libSVM [3] library (written in C) was chosen. To use all the cores of the processor, openMP [6] for libSVM is enabled.

Table 3.3 shows the performance of the software implementation on a 2.66 GHz 6-core Xeon X5650 with hyper-threading enabled and 12 GB of DDR3 RAM. It shows that the software implementation is not able to support 10 Gb/s. The best supported speed ranges is from 2.32 Mb/s to 1597 Mb/s depending on the trace.

- The flow reconstruction speed does not depend on the trace as the flow reconstruction algorithm requires a constant time per packet. The only noticeable

difference is for the biggest trace, FT, where the flow reconstruction probably suffers from the heavy CPU usage of the SVM classification.

- SVM classification is always more limiting than flow reconstruction (0.024 % of the requirements for 10 Gb/s in the worst case). Its speed depends on different factors including the number of support vectors in each SVM model: Brescia is the trace for which the learnt model has the most support vectors (24 758), then come FT (6 296) and Ericsson (4 341).

Trace	Packets per second (flow reconstruction)	Flows per second (classification)
Ericsson	5 189 293 76 % of 10Gb/s req.	4 655 17 % of 10Gb/s req.
Brescia	5 153 675 9.2 % of 10Gb/s req.	1 031 0.40 % of 10Gb/s req.
FT	4 336 677 9.0 % of 10Gb/s req.	311 0.024 % of 10Gb/s req.

Table 3.3: Performance of the software implementation compared to 10Gb/s requirements

Even with a powerful computer, a software implementation is not able to reach a 10 Gb/sec. speed, mainly due to its limited ability to parallelize the computation. This justifies the use of hardware acceleration. We will report now the implementation of a hardware accelerated version of the detection phase of the SVM algorithm, thus corresponding to Equation 3.1.

3.3.2 Requirements for the hardware accelerated classifier

The traces used to test the classification algorithm are described in Table 3.1. The average sizes of packets and flows in bytes vary for these traces. Requirements in terms of packets/sec. and flows/sec. supported by the algorithm to reach a 10 Gb/s speed are described for each trace in Table 3.4. To support each trace sent at 10 Gb/s, the flow reconstruction should support at least 55 861 124 packets/sec. and the SVM classifier should support at least 1 279 231 flows/sec.

Trace	Packets per second	Flows per second
Ericsson	6 809 840	27 805
Brescia	55 861 124	256 472
FT	48 310 718	1 279 231

Table 3.4: Requirements for a classification at 10 Gb/s for each trace

3.4 Design of the hardware accelerated classifier

3.4.1 Fixed point data representation

Floating-point operations are complex to realize in hardware and use too much area on the FPGA. The best solution is to transform the algorithm to use a fixed-point model instead of a floating-point model. Table 3.5 shows the bit widths of different variables used in the SVM fixed-point model.

Variable	Integer part	Decimal part
Vector component	11	0
α	7	11
γ	0	18
b	15	11
S	15	11

Table 3.5: Quantization of the main SVM fixed-point model values

These quantization parameters have been chosen so that the mathematical operations are made on values as small as possible, without loosing too much precision for the classification. Some sizes are quite large because the classifier should work whatever the SVM model, so that a new synthesis is not required to change the model. The possible values of the variables have been determined by analyzing SVM models learnt in different conditions. For example the precision of the γ parameter is very important (decreasing it leads to a drop in classification accuracy), but its absolute value never reaches 1. The 11-bit width of a vector component has been chosen because we assume that the size of a packet will not be more than 1500 bytes.

Multiplications are complex to realize in hardware. They are required to compute the squares in the kernel, but squares are symmetric functions with one integer parameter varying from -1500 to 1500. A ROM with 1501 values is used to emulate squares. Similarly, a ROM is used to emulate the exponential function. Finally, to avoid the $y_{d,i} \times \alpha_{d,i} \times k_i$ multiplication, $ln\left(|y_{d,i} \times \alpha_{d,i}|\right)$ is precomputed, and the exponential used to compute k_i is computed only after the addition of this term. Delaying the exponential computation transforms the multiplication into an addition. This way only one multiplication by a constant remains in the kernel computation, which is much simpler than a multiplication of two variables.

To check that the loss in precision is not too important, a software implementation of the classification algorithm with the fixed-point model has been implemented. Figure 3.2 compares the accuracy of the fixed-point model to the results of the float model. It shows that the transition to fixed-point decreases the accuracy of the algorithm, but it remains around 90 %. Depending on the requirements, a higher accuracy can be achieved using wider fixed-point values, but it will require more space on the FPGA.

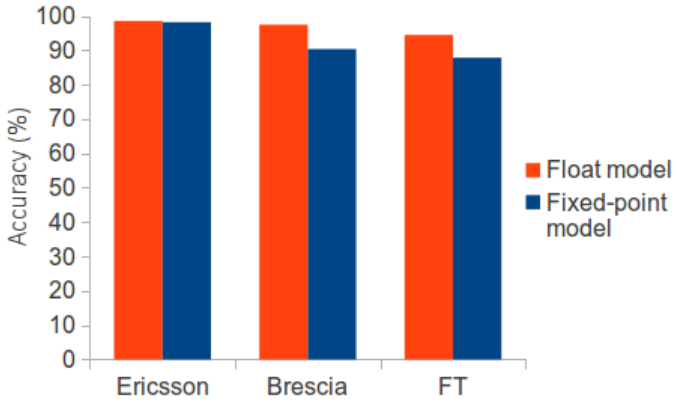


Figure 3.2: Accuracy of the fixed-point model compared to the float model

3.4.2 Hardware architecture

The architecture of a traffic-processing module on NetFPGA or Combov2 cards is very similar. It uses a block with an input bus for input traffic, and an output bus for output traffic. The classifier block is described in Figure 3.3.

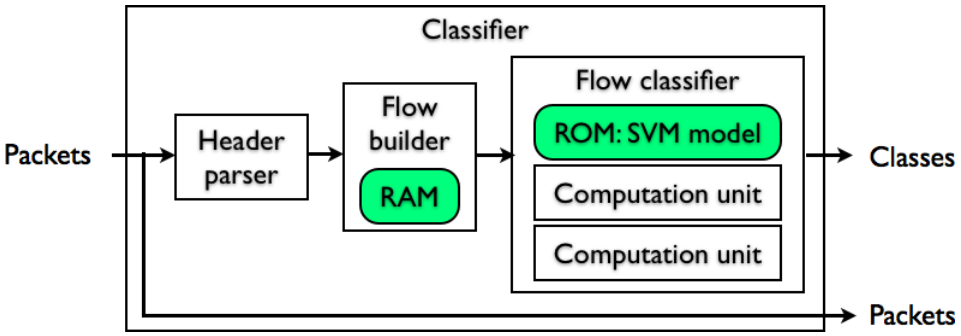


Figure 3.3: Architecture of the classifier

The computation units represent the most important part of this architecture: they implement the computation of the main loop described in Algorithm 1. To get the best performance from the FPGA, operations of the algorithm must be parallelized. All loops can be totally unrolled by duplicating the hardware for each iteration except the main loop. The computation unit is duplicated as much as the Virtex-5 supports.

As the computation in the main loop is complicated, each iteration will take many clock cycles in hardware. To improve the throughput of the loop and reduce its computation time, the iterations can be pipelined: one new support vector is processed by the first operation of the computation unit at each clock cycle, and then forwarded to the next operation. This way all operations work in parallel and each computation unit accepts one support vector at each time step.

3.4.3 Synthesis results

Synthesis results with a RBF kernel

Trace	Ericsson			Brescia		FT		
Computation units	2	4	8	2	8	2	4	8
Occupied slices	8 414	14 186	26 350	24 340	32 679	10 174	15 643	26 846
Occupied slice registers	9 221	21 864	45 967	9 272	40 658	8 966	21 287	44 356
FPGA usage (% of slices)	22.47	37.89	70.38	65.01	87.28	27.17	41.78	71.70
Maximum frequency (MHz)	174	156	165	51	62	157	164	139
Cycles per flow	2 193	1 110	569	12 401	3 121	3 170	1 598	813
Flows per second	79 733.6	140 766	290 107	4 122.42	20 164.3	49 741.0	102 840	171 861
% of 10Gb/s requirements	286.8	506.3	1043	1.61	7.862	3.888	8.039	13.43

Table 3.6: Synthesis results of SVM traffic classification on a Virtex-5 XC5VTX240 FPGA

To assess the performance of the hardware implementation and compare it to the software implementation, it has been synthesized on a Virtex-5 XC5VTX240. Three different SVM models (one for each trace) have been tested. The number of processing units has been changed as well, to exploit the maximum parallelism on the FPGA. Table 3.6 presents the results of these synthesis.

The number of occupied slices and slice registers as well as the maximum frequency are given by the synthesis tool. They are an indication of the hardware complexity of the implementation. The number of cycles required per flow has been determined by analyzing the code of the hardware implementation. It increases with the number of support vectors in the model, and decreases with the number of parallel computation units.

Thanks to massive parallelism, hardware implementations all have better performance in terms of flows per second than software implementations. The implementation for the Brescia trace gives poor results because of its low working frequency. The particularity of this trace is that the SVM model contains more support vectors than the others. They use too much space on the FPGA, which creates long and slow routes in the design an decrease its maximum frequency. The Ericsson and FT traces SVM models have less support vectors. Even with only 2 computation units, the implementation for the Ericsson trace gives results much higher than the requirements to support a 10 Gb/s speed (286 % of the requirements). The implementation for the FT trace brings roughly the same performance improvements, but the requirements in terms of flows per second are very high because the trace

contains many very small flows. So with 8 computation units, it fulfills only 13 % of the requirements.

Synthesis results with a CORDIC kernel

The computation of the kernel and its repetition for all support vectors is a ressource consuming step in the SVM classification algorithm. Rather than using a RBF kernel it is possible to use a CORDIC algorithm for simplifying the computation of the kernel [8]. The CORDIC kernel is more adapted to hardware implementation of the SVM algorithm than the RBF kernel. From our first comparisons in software the classification accuracy is higher with the CORDIC kernel than with the RBF kernel. The comparison of the performance offered by both kernels in terms of the maximum number of flows classified per second in hardware is still ongoing work.

3.5 Validation on generated traffic and on a campus network

The last step of this study is to go further than synthesis results that is to say to validate the hardware accelerated traffic classifier on some high bit rate traffic. To this end we have followed two directions in parallel:

- The first direction is to validate the classifier on some traffic which is generated either by a commercial traffic generator such as the XENA generator that we have borrowed from the Infractive company. As we do not own such a traffic generator we have also designed and implemented a hardware accelerated traffic generator in the framework of a project with Telecom Bretagne students. Two versions of this generator have been designed, one for the NetFPGA 1G board and the other one for the COMBO board.

- The second direction is to validate the classifier on a production network. In order to do so we have signed an agreement with the association of Telecom Bretagne students in order to be authorized to monitor the traffic of the students' dormitory for research purposes. A NetFPGA 1G card has been installed behind a mirroring port of the router of the students' dormitory in order to test the behavior of the classifier in a more realistic environment in terms of trafic composition.

3.5.1 Design of a hardware accelerated traffic generator

To date it is not possible to generate trafic at several Gb/sec. or tens of Gb/sec. with a software approach on a commodity computer. The main limitation is the capacity of output ports of the Network Interface Card of the computer, as well as the capacity of the computer to coin packets at very high speed. A possible solution would be to coin packets with a software approach on the computer and then send them to the NIC which stores them and replicates them on its output port. It is impossible to generate trafic with this approach at a bit rate higher than 1 Gb/se because of the limitation of the PCI bus that connects the NIC to the computer.

In order to test our hardware accelerated traffic classifier it was then necessary to use a high bit rate traffic generator such as the Agilent, Spirent or XENA traffic generators and analyzers. As we do not own such an equipment we have borrowed a XENA generator from a vendor. Another more challenging option has been to design and implement our own traffic generator on a FPGA board. We have used the NetFPGA and COMBO boards and we have designed and implemented different versions of hardware accelerated traffic generators in the framework of projects and internships with students at TELECOM Bretagne. The goal was to design a flexible hardware accelerated classifier able to produce traffic with a range of speed between 1 Gb/sec. and 20 Gb/sec. using either the NetFPGA 1G (4 interfaces at 1 Gb/sec.) board of the COMBOv2 board (2 interfaces at 10 Gb/sec.).

In our design the generation is performed in hardware on the FPGA board and the board is controlled by the host computer. A software module has been implemented in order to permit the specification of flows and the transfer of flow specifications to the FPGA board. Flow treatments as well as packet generation is then performed on the FPGA board in order to benefit from hardware acceleration. A flow specification format has been decided for; flows are specified by the end user in a XML file which is parsed in Python by the host computer. The specification of each flow is then sent to the FPGA board where it is stored in memory.

A specification reader reads randomly a flow specification from memory, then generates a given number of packets on the basis of this specification read and rewrites the specification in memory after having taken into account the number of packets of this flow that have been generated. A packet generator builds TCP or UDP packets by coining the headers of the different layers of the TCP or UDP stack. Eventually a packet sender sends the packets that have been generated on one of the output serial ports of the FPGA board.

3.5.2 Validation on a campus network

In parallel with the studies on high bit rate traffic generation another research line has been to set up a probe on the Résel. Résel is the acronym for "Réseau des Elèves". This network provides Internet connection to the students dormitory (approximately 600 students) at Télécom Bretagne.

P2P is prohibited by Renater since this traffic consumes some bandwith and is often used to exchange copyrighted material. For that reason the administrators of the Résel have to ban this traffic. We have considered a scenario in which P2P traffic is detected thanks to the hardware accelerated SVM based traffic classifier. Then, upon detection, IPTABLES firewall rules are modified in order to block P2P traffic. A high level view of the architecture is depicted on picture 3.4.

The probe which is based on a NetFPGA 1G board is located behind a mirroring port of a router. It is necessary to store the SVM model in the memory of the NetFPGA board. The SVM model has been trained thanks to the GT tool of the university of Brescia as it is depicted on Figure 3.5. The detected applications are BitTorent, Gnutella, FastTrack, eDonkey, GNUnet, Direct Connect and "Others" which represents the rest of the traffic.

The architecture is operational and efficient in a laboratory context. Our system

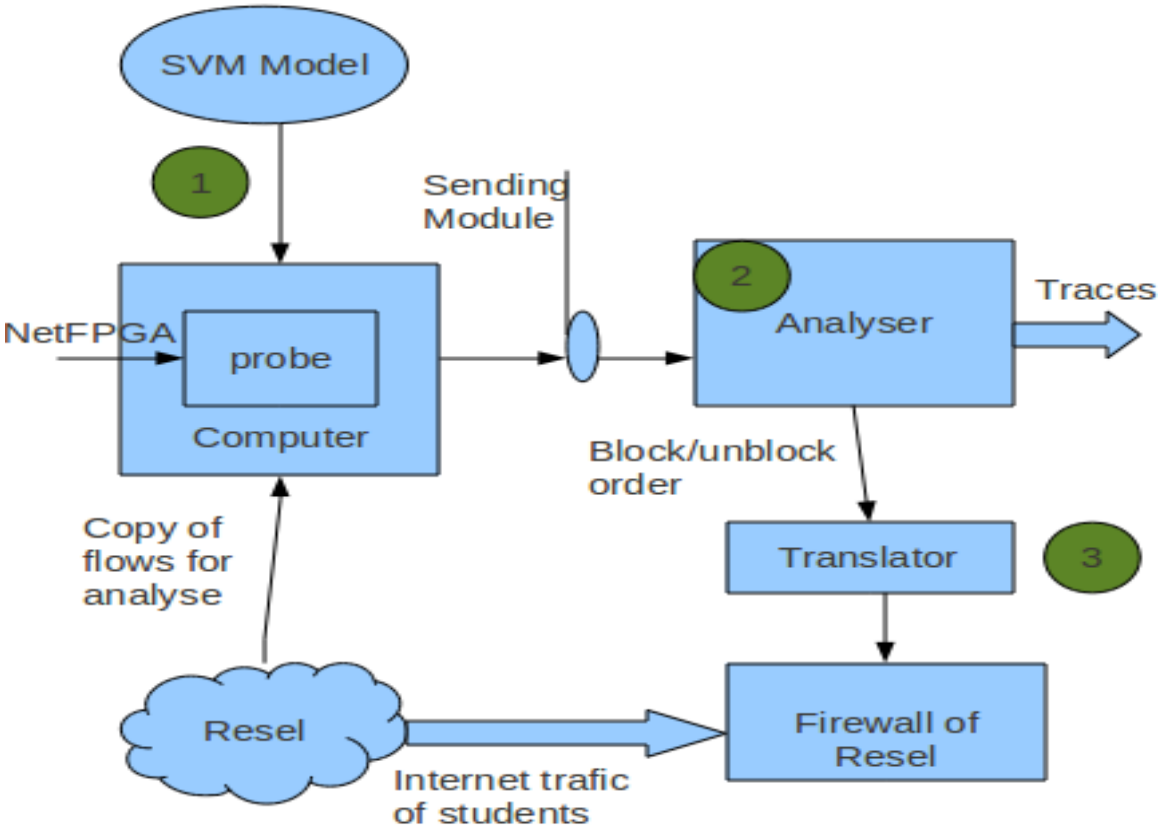


Figure 3.4: Monitoring probe on the student's network

is able to detect and block appropriately P2P traffic with no observed false alarms or misses provided that the SVM model has been trained appropriately. For the moment the system is not operational on the Résel where false alarms have been observed. The conclusion of this study is that more efforts must be put on the design of a flexible and reliable learning methodology for a supervised traffic classification method such as the SVM one. Also it is necessary to speed up the learning phase of the SVM algorithm.

The learning phase is done offline and before the detection phase. It can take a considerable computation time as it must find the best hyperplanes that separate classes of traffic by doing multiple tests. It has the advantage of being done at a very low frequency compared to the detection phase; typically traffic signatures should be changed at a monthly time scale. Although this phase can be performed off line it is CPU intensive and this is why we have accelerated the learning phase of the SVM based traffic classification using the Graphical Processing Unit (GPU) as it has been previously described in deliverable D2.2. (section 2.4.3).

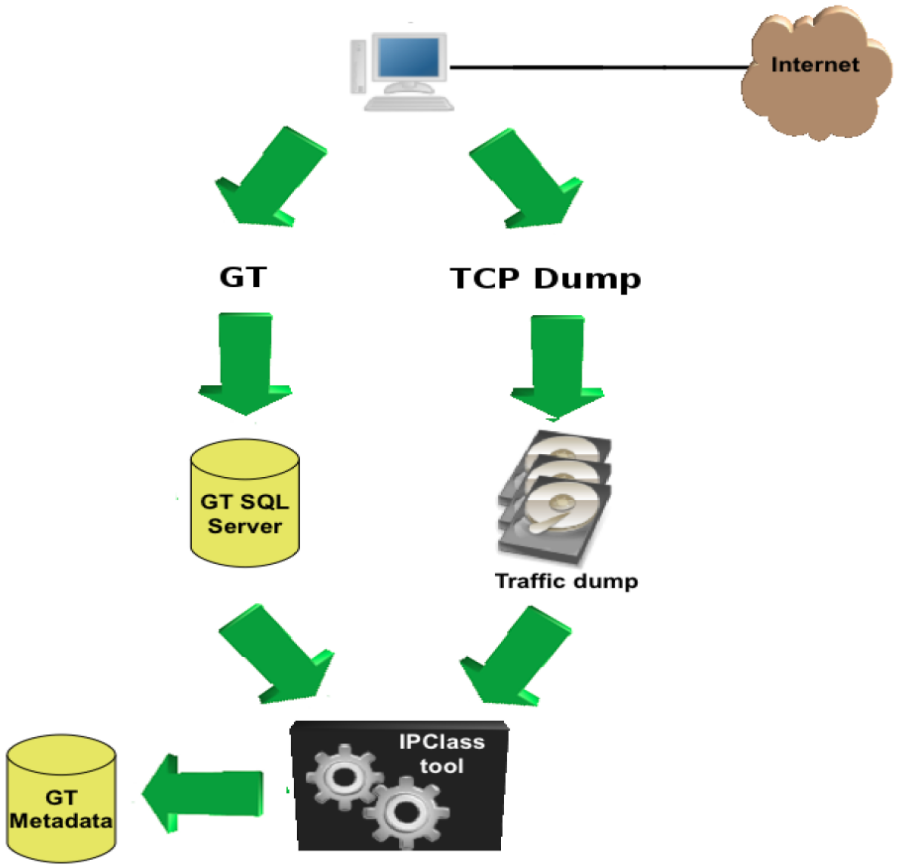


Figure 3.5: Learning step

3.6 Conclusion

In this chapter we have summarized the main advances of our researches on traffic classification. The conclusion of these studies is that it is possible with FPGA boards to implement SVM based traffic classifiers able to process a traffic rate up to tens of Gb/sec. The performance of SVM based traffic classification in terms of accuracy is excellent in a laboratory environment. But we believe that this method is still not enough reliable to be deployed in an operational network. Its main weakness is that it requires an accurate traffic model which is difficult to obtain in a highly heterogeneous and changing environment although. We believe that additional efforts should be performed in this direction.

4 Integration of Monitoring Metrics in the VIPEER Demonstrator

4.1 Introduction

The VIPEER demonstrator deploys an ICN-like intra-domain video streaming architecture allowing to minimize the load of peering links and to have better quality of experience at the client side. For this, a set of servers dedicated by the ISP cache chunks of the videos (dCDN Servers) allowing clients to retrieve chunks from the most appropriate server. These servers form a dCDN (a Distributed Content Delivery Network).

As the solution is based on DASH (dynamic adaptive Streaming over HTTP), each chunk is an HTTP request. DASH enables high quality streaming of media content over the Internet delivered from conventional HTTP web servers.This mechanism works by breaking the content into a sequence of small HTTP-based file segments. Each segment contains a short interval of playback time, typically about 2 seconds, of a content that is potentially many hours long. The content is made available at a variety of different bit rates covering a range of quality, which could be from very bad to high quality. This mechanism allows a user to have the best quality possible regarding network troubles. This request arrives to a redirection server located at the dTracker machine. The latter machine is the intelligence of the system. It decides which is the best server for the current chunk according to the current measurement results provided by the measurement components of VIPEER. In case a server is selected, a redirection is sent to the client with the URL of the chunk at the selected server.

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 are 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

(i.e. available bandwidth > video degraded rate), select the nearest server. Again the metric used for this selection is the RTT (round-trip time).

3. Redirect the client to the original CDN server.

Figure 4.1 plots the server selection strategy at the dTracker level.

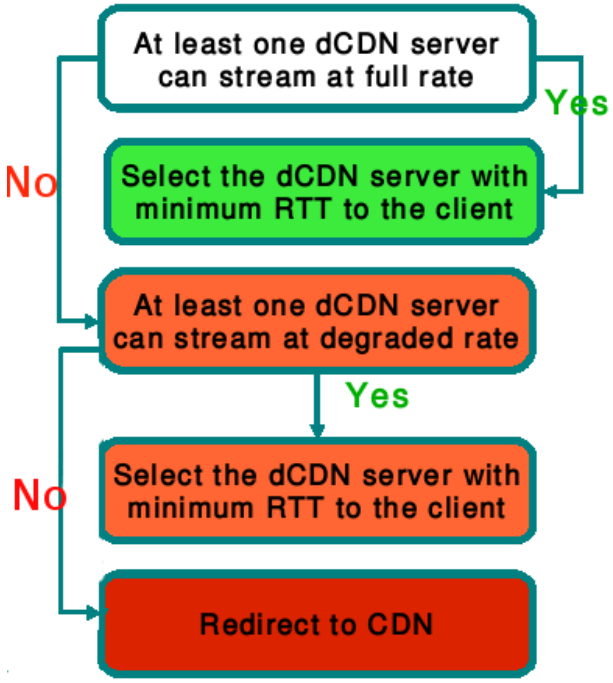


Figure 4.1: dServer Selection strategy at dTracket level

In the following paragraphs, we describe the measured metrics and the measurement architecture deployed in the VIPEER demonstrator.

4.2 Metrics

4.2.1 Round-Trip Time (RTT)

As HTTP is used as a support of data exchange, it is worth to optimize round-trip time between clients and streaming servers. RTT can be measured by using simple ICMP-based ping messages. In the implementation, servers are continuously pinging the clients currently connected to the system in order to measure the current values of the round-trip time. On the decision level, a weighted moving average of the RTT has been implemented in order to take into consideration transient effects and to smooth the evolution of the global system.

As this method is an active one, it can be too much bandwith demanding in a huge network. In order to solve that problem one could imagine having a representative client for each geographic zone thus limiting the measurement traffic. In order to limit the RTT measurement traffic, the system has been configured in our experiments as follows:

- On one hand, when a Ping process is started from a server to a client, it stops when it receives the first ICMP Echo Reply message. This limits the number of messages exchanged during a measurement as it can not last more than an RTT.

- On the other hand, the period of measurements is taken equal to 10 seconds that is to say the duration of 10 chunks.

As a consequence these simple rules have made it possible to limit the length and rate of bursts in the measurement traffic.

4.2.2 Available upload Bandwidth

As dCDN Servers have limited upload capacities, it is worth distributing the load of streaming among them. It is important to monitor the current upload rate of these nodes in order to decide whether to attribute new streaming jobs to them or not. It is very easy to compute the available upload rate by substracting the current upload rate to the known maximum upload capacity of servers. The latter is measured by looking into the statistics of the network interfaces using the *netstat* command.

4.3 Measurement architecture

As the dTracker will take redirection decisions, it is necessary that a global view of the state of servers and network conditions is centralized in a database accessible by the dTracker. This monitoring database is managed by a measurement controller that orchestrates all the measurements. Measurement nodes located on dCDN Servers are constantly in contact with the measurement controller in order to receive measurement orders and to send back measurement results.

In the following paragraphs, we detail the roles and functionalities of the measurement controller and of the measurement nodes.

4.3.1 Measurement Controller

The Measurement Controller is the controller unit which communicates with all the measurement nodes located in the dCDN servers.

Measuring RTT

This Controller detects whenever any new client gets connected to or disconnected from the streaming system:

- Whenever any new client gets connected, the Controller sends a message to all measurement nodes in order to inform them that there is a new client, and ordering them to start measurements of RTT (Round Trip Time) to this client.

- Whenever any client gets disconnected, the Controller sends a message to all the measurement nodes that this client is no more connected. So, it orders them to stop measuring RTT to this client.

RTT is being measured using the "ping" command. Measurement nodes send RTT results to the Controller. Whenever the latter receives RTT from any measurement node for any specific client it updates the current RTT estimate as ($RTT = measuredRTT * 0.25 + RTT * 0.75$).

The Controller is continuously reading the clients and servers tables in the system's database to detect whether their states have been updated to the states CONNECTED or DISCONNECTED. A newly connected streaming client is detected when it sends a new request for a chunk. In case it has been inactive for 60 seconds, we consider that the client is disconnected.

Measuring the Available Upload Rate

Measurement components at dCDN Servers are periodically sending the current upload rate to the measurement controller in order to compute the available upload bandwidth ($AvailableUploadRate = MaximumUploadCapacity - UploadRate$).

For experimental reasons, it is possible for the measurement controller to throttle the bandwidth of any of the dCDN Servers. It then sends a message to the concerned measurement node only. 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 experimentations with "overloaded" servers.

4.3.2 Measurement Node

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 the Current Upload rate using the "tc" command. This "tc" command is used to configure Traffic Control in the Linux kernel. It consists of shaping, scheduling, policing, dropping.
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 10 seconds 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, the client 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 the "tc" command.

4. Send back results of measurements periodically to the Controller.

4.4 Example of measurement results

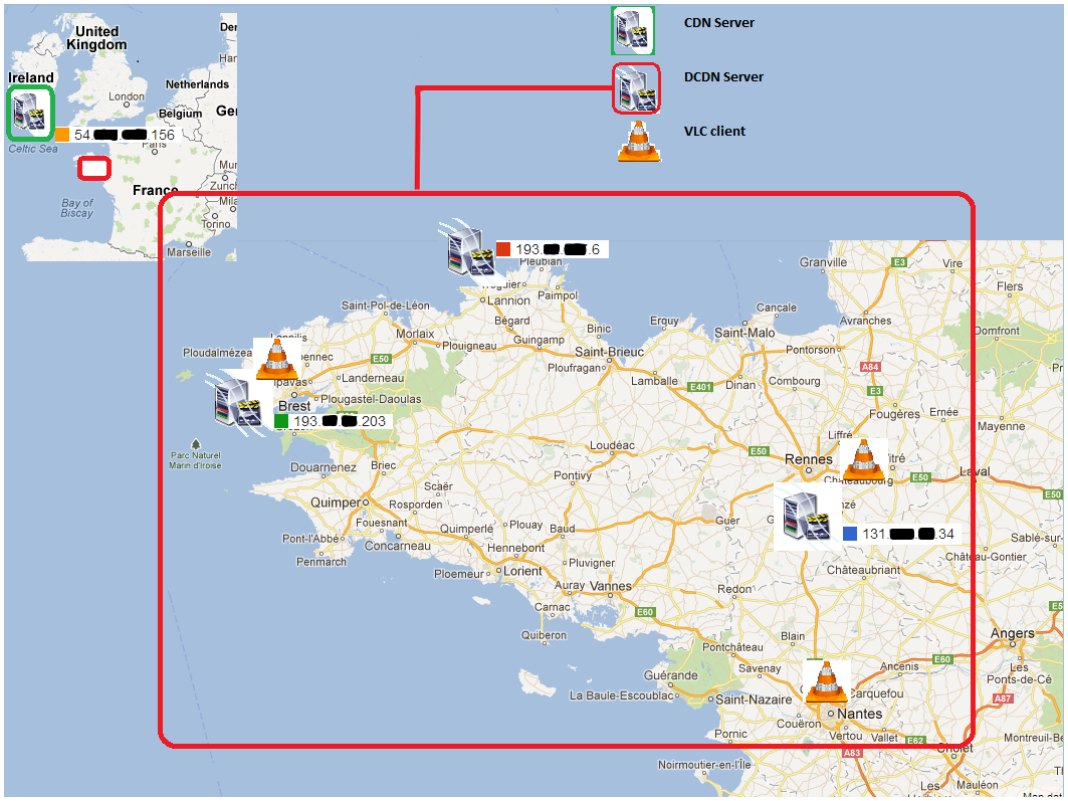


Figure 4.2: Testbed architecture

A distributed inter-partner platform has been installed to demonstrate the functionalities of the global streaming system (see Figure 4.2). Table 4.1 gives an idea about the machines used in the demonstrator. In our experiments, we have deployed some dCDN Servers at partners sites (one dServer at Telecom Bretagne, Brest, another at INRIA, Rennes, another at Orange Labs, Lannion). A dedicated server has been rented by NDS Limited at Amazon Ireland in order to play the role of the original CDN server. dTracker and measurement controller have been deployed at Telecom Bretagne, Brest. An additional Nantes client appears in Figure 4.2, it has been used for a demonstration during the *Loading the future demo trophy* organized by the *Pôle Image et Réseaux* in Nantes, France. Figures 4.3 and 4.4 plots respectively the RTT to different clients from different servers and the available upload rates at these servers.

Table 4.1: 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

Figure 4.3 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. In this experiment, each client is streaming four videos in parallel at a rate of 3000 kbit/s each. 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 (dark green curve). 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 (cyan blue curve).

The second monitored metric is the available bandwidth at servers. It is shown in Figure 4.4 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

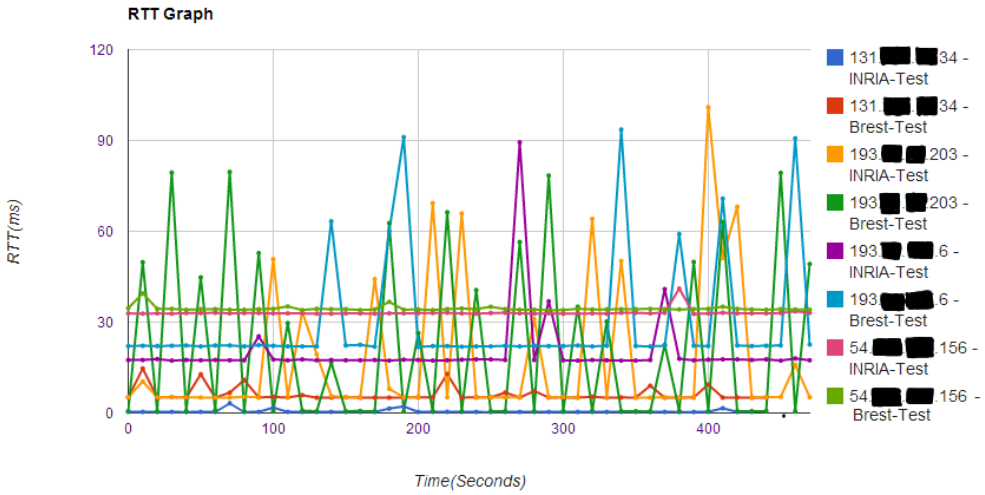


Figure 4.3: RTT

which is one of the main goals of VIPEER.

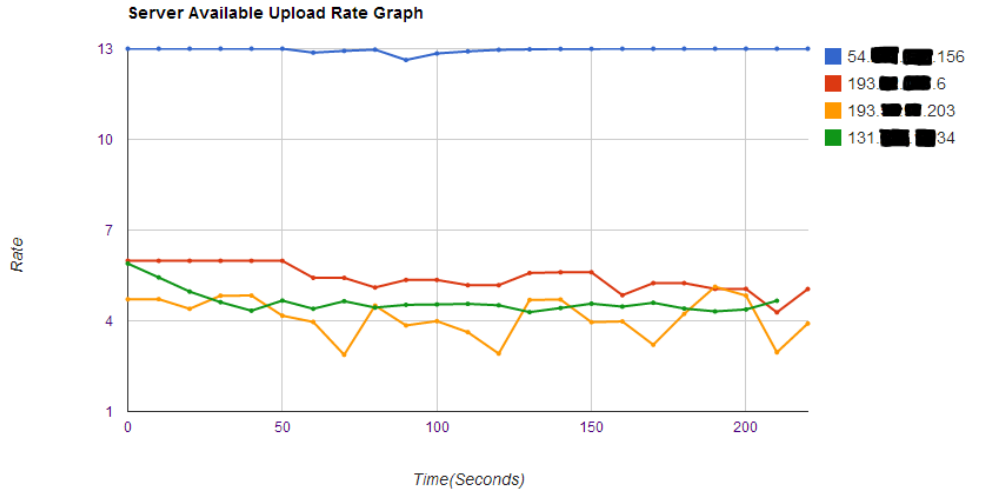


Figure 4.4: Available upload rate

4.5 Conclusions and Future work

Measurement components integrated in the VIPEER demonstrator allowed to provide metrics for server and chunk selection. The current work aims at integrating the monitoring of the QoE perceived by the clients in the VIPEER prototype. For additional information on the VIPEER prototype please refer to D5.5.

5 Conclusion

We presented in this deliverable the final technical specifications of the tools and modules proposed within the WP2, aka traffic classification, QoE estimation and network metrics. Regarding traffic classification, it is notable that with FPGA boards it is possible to implement SVM based traffic classifiers able to process a traffic rate up to tens of Gb/sec. The performance of this method in terms of accuracy is excellent in a laboratory environment but an accurate traffic model is still needed for an operational network. Concerning the QoE estimation, the proposed solution consists in taking into account the variation of the video quality over different chunks. These DASH-based video streaming techniques are predominant ones today in streaming over Internet. The result of this work consists in measuring the QoE of DASH-like video streaming and to predict the impact of a quality degradation on the perceived quality. These two topics are not implemented in the final demonstrator:

- for the QoE estimation it is due to a lack of time but an independent demonstrator is available.
- for the traffic classification it is due to a change of concept: the first idea was that content could be stored in set top boxes and so the knowledge of user activity was needed to manage free user capacity in real time. Finaly, works in WP4 are more oriented in storage on servers (dCDN servers) than in set top boxes, and results show that this kind of storage is suitable.

Finally, the network metrics components are integrated in the final demonstrator and they provide metrics for the selection of suitable server and chunk.

Bibliography

[1] ITU-R Recommendation BT.500-11. Methodology for the subjective assessment of the quality of television pictures, 2002.

[2] CESNET. Our hardware. http://www.liberouter.org/hardware.php?flag=U, November 2011.

[3] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[4] Clear Foundation. l7-filter: application layer packet classifier for Linux. http://l7-filter.clearfoundation.com/.

[5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[6] L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. *Computational Science Engineering, IEEE*, 5(1):46 –55, jan-mar 1998.

[7] I. Daubechies. *Ten Lectures on Wavelets*. Number 61 in CBMS/NSF Series in Applied Math. 1992.

[8] J. Gimeno Sarciada, H. Lamel Rivera, and M. Jiménez. CORDIC algorithms for SVM FPGA implementation. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7703 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, April 2010.

[9] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K.C. Claffy. GT: picking up the truth from the ground for Internet traffic. *ACM SIGCOMM Computer Communication Review*, 39(5):13–18, 2009.

[10] NetFPGA. NetFPGA: a line-rate, flexible, and open platform for research, and classroom experimentation. http://netfpga.org/.

[11] Yen-Fu Ou, Zhan Ma, Tao Liu, and Yao Wang. Perceptual quality assessment of video considering both frame rate and quantization artifacts. *IEEE Trans. Cir. and Sys. for Video Technol.*, 21(3):286–298, March 2011.

[12] K.D. Singh, Y. Hadjadj-Aoul, and G. Rubino. Quality of experience estimation for adaptive http/tcp video streaming using h.264/avc. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pages 127 –131, jan. 2012.

[13] T.Groléat, M.Arzel, and S.Vaton. Harware Acceleration of SVM-Based Traffic Classification on FPGA. In *IEEE IWCMC Conference, TRAC Workshop*, 2012.

[14] M. Žádník and L. Lhotka. Hardware-accelerated netflow probe. Technical report, Technical Report 32/2005, CESNET, Praha, 2005.