Metrics for QoS in Real-Time Interaction over the Internet

Christophe Salzmann, Denis Gillet and Philippe Müllhaupt
School of Engineering, EPFL - Swiss Federal Institute of Technology
Lausanne, Switzerland
first.lastname@epfl.ch

ABSTRACT

Real-time Interaction over the Internet (RTI2) is an Internet service that is required typically by remote experimentation applications. From a quality of services (QoS) point of view, RTI2 has constraints that differ from usual real-time multimedia services such as video streaming or video conferencing. The RTI2 QoS can be expressed by three values that represent the level of interaction, the perceived dynamic and the semantic content. The RTI2 metrics, derived from these values, are essential to successfully implement an end-to-end (E2E) control scheme that adapts the transmission parameters not only to the network state, but also to the server and client applications processing capabilities. A macroscopic view of the system is exploited to ensure applications, protocols and infrastructures independence.

Keywords: Metrics, real-time, Internet, interaction, Quality of Service, End-to-end approach

1. INTRODUCTION

Context and aims

The key issue in implementing real-time interaction over the Internet with a physical system is to enable its control and the perception of its dynamics at distance. Physical systems under study are typically mechatronic systems with moving parts. These systems are challenging because their intrinsic time constant are generally in the same order of magnitude as the internet transmission time. RTI2 has been extensively used for remote experimentation of systems such as robots, inverted pendulums, electrical drives, etc. (Fig. 1) These systems are typically used in laboratory where students can access them either locally or asynchronously in both time and place [1].

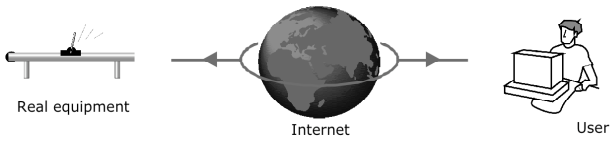

Fig. 1. Typical remote experimentation setting

Since interaction is required, the need for the operators to get feedback as quickly as possible on actions carried out is a constraint that requires dedicated solutions. RTI2 aims at providing the user with the best possible feedback such that he/she is not disturbed by the drawbacks inherent of the distance. There are three key aspects that need to be satisfied in order to provide a suitable quality of service, namely the level of interaction which represents how quickly a feedback is provided to the user; the perception of the dynamics which represents how accurately in time the behavior of the remote system is perceived and the amount of semantic content that represents how much of the distant system state and conditions of operation can be perceived by the client. The applications, protocols and infrastructures independence is also required to enable an easy deployment. The component independence indicates that the proposed approach works over a wide range of applications, hardware and network configurations and that it will adapt to future versions of these components. The fair bandwidth usage implies that the comportments of the solution are compatible with the Internet best practices [2].

Current implementation solutions

Various solutions can be explored to efficiently implement real-time interaction over the Internet. While the video streaming solution looks suitable for RTI2, the use of buffers to smooth the Internet bandwidth variation and to display images at a constant rate to the user makes it inappropriate since the buffering process add delays to the transmission [3]. Video conferencing is another real-time application that seems comparable to RTI2 but it carries differences: the priorities for the video and the audio are inverted. Sound is preponderant over image for video conferencing while this is not the case for RTI2. There might even be no sound at all. In video conferencing the amount of data transferred between the two parties is generally symmetrical, this is not the case for RTI2 where only a small amount of data goes from the client to the server but a large amount of data flows from the server to the client. Another difference is the scalability of the used bandwidth. RTI2 bandwidth usage ranges from a few bytes per second to Kilobytes per second. The former bandwidth corresponds to a client application running on a PDA with a Bluetooth network access and the later correspond to a client running on a desktop computer with a LAN access. The lowest values cannot be considered for video conferencing due to sound quality constraints.

A straightforward solution to implement RTI2 is to use a communication channel that can guarantee a given quality of

service [4], such as a given bandwidth and latency, via reservation or by other means. This can be done by placing additional intelligence in the network at the router level. While this solution might be a promising one, it not only requires a widely accepted agreement among manufacturers and providers regarding new communication protocols, but also asks for expensive software upgrades for most of the already deployed infrastructure.

Instead of trying to modify the routers behaviors, the proposed solution is based on an end-to-end scheme that can be implemented at the application level. Proposed Internet improvements such as differentiate services, bandwidth reservation, packets coloring, etc. are not generally available and therefore not considered *a-priori,* but the proposed approach implicitly take advantage of them when available.

Main results

The three key aspects of RTI2 – the level of interaction, the perception dynamic and the information semantic content – that define a efficient user experience can be translated in three controllable quantities – the information transmission delay, the information delivery pace and the information size. The basic unit of information that transits from the server to the client application is defined as a block. The metrics used to evaluate the achieved QoS are the ratio of the perceived block pace and block size to the initial block pace and block size. The last metric being the block E2E round trip time measured at the application level.

Paper organization

The component and block abstractions that are used to implement the end-to-end approach are first defined. Based on these abstractions, the metrics are then defined. The paper concludes by providing experimental results that enlighten the advantage of the proposed metrics compared to bandwidth only measurement.

2. END-TO-END APPROACH AND ABSTRACTIONS

The proposed approach includes not only the transmission path along the Internet but also the client and server applications located at both ends. This End-to-End approach characterizes the overall transmission path, from the information capture to the information rendering. The inclusion of both the server and the client applications is required by the metrics measurements that reflect not only the transmission path characteristics but also the client application ability to handle the flow of information coming from the server application.

The client application is meant to run on various unknown before hand computer devices from personal digital assistants (PDA) to desktop computers that are connected to the Internet with the help of a wire or wireless connection. Since the network does not necessarily act as the bottleneck anymore, the characteristics of the client application are to be taken into consideration so as to prevent the server application wasting network resource by sending more information that the client can process.

A macroscopic view of the E2E transmission path abstracts the client-server application and the network transmission into components (fig. 2). This abstraction frees the solution from the underlying constrains such as the transmission protocols, the software, the hardware and the network configuration. It also permits the components adaptation to future versions of these underlying constrains.

Components abstraction

The information undergoes various transformations from the client application to the server application that can be abstracted into three components, namely the acquisition–encoding component, the transmission component and the decoding–rendering component.

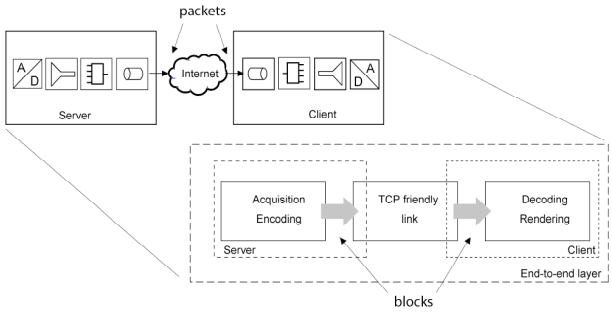

Fig. 2. End-to-end layer components abstraction

The **acquisition-encoding component** transforms the system state and its conditions of operation to its digital representation. This is mainly done via data acquisition devices. One of theses devices can be a camera that would produce an image. The image can then be compressed to reduce the size of the transmitted data. Another source of information is the measurements made via a data acquisition (DAQ) board. The various flows of information will be aggregated to form the basic information unit called block.

The **decoding-rendering component** is very similar to the acquisition and encoding component. The information simply goes the reverse path, first the block information flows extraction, then the information decompression and finally the information rendering.

The **transmission component** encompasses the sever network interface and the client network interface. The transmitted information is handled by this component as soon as the control of the transmitted data is transferred from the server application to the underlying OS. The transmitted information leaves this component when the client application has access to this information. There is no handle to control the transmission over the Internet once the data leaves the computer and until it is received at the other end. This is due to the non-deterministic aspect of the best effort Internet network and to the nature of the protocol used to transmit the data. The routers along the transmission path simply do their utmost to deliver the data to the receiver as fast as possible despite the variation of the network load. In other words neither the network bandwidth nor the network latency can be guaranteed.

A **block** is defined as the aggregated information that represents the state and the operating conditions of the distant system at a given time. It is the basic unit of information for the semantic

content. For example, a block is made of a video image combined with the measurements acquired concurrently (Fig. 3). Blocks of various sizes and various periods are generated by the acquisition-encoding component.
Blocks need to carry additional information to permit the real-time information processing and playbacks in the decoding-rendering component, this even if blocks were partially or completely lost during the transmission. These additional information, stored in the block header, are the block identifier, the block timestamp and the block period.

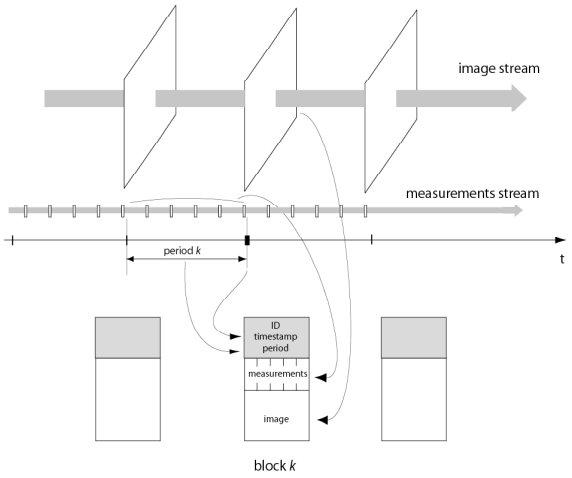

Fig. 3 block abstraction and content

Block content
The beginning of a block starts with a unique value. Special care it taken to ensure that this value cannot be found in any other part of the transmitted data. This mark is used by the client application to re-synchronize the receiving process when losses occur. The block **timestamp** represents the server time at which the block has been created. This timestamp defines when the block has to be processed at the client side to ensure the real-time block playback. The block **period** represents the elapsed time between the current block creation and the next block creation. The block period is the inverse of the current block pace. The block **ID** incremented by one unit every block, is used to determine if one or more blocks where lost during the transmission. The block timestamp and block period carry redundant information since the current timestamp plus the period should be equal to the timestamp of the next block. This assertion only holds if there is no block loss during the transmission. If there is one or more lost blocks, the block timestamp and block ID will be used to render adequately the newly received block.

A block might be split into packets when transiting over the Internet. Depending on the protocol used for the transmission, the block reconstruction might be built-in the protocol or might require additional intelligence at the receiver to reassemble the packets into a block. In order to simplify the block reconstruction at the client side, if this is not handled by the underlying protocol, each packet needs to carry a replica of the information contained in the block header. The assumption that packets will not be split along the transmission path can be guaranteed by splitting block in packets of size smaller or equal to the path MTU.

Block is not to be confused with packets, smaller data units transmitted over the network and handled by the communication protocol.

3. METRICS FOR QoS IN RTI2

The efficiency of the user experience can be expressed by three key aspects – the level of interaction, dynamic and semantic content.

The level of **interaction** can be characterized by the delay observed between the time an action is performed by an user and the time its effect can be visualized by the user. The delay represents the round trip time measured at the application level. This delay is function of many factors, especially buffers that are found along the transmission path. For a valuable user experience, the delay should be as small as possible, and should also be in accordance with the dynamics of the distant system. Systems with slower dynamics show less stringent constraints. If this delay cannot be kept to a minimal value, the experience is deteriorated, special care in the solution design may partially compensate for a slightly excessive delay.

The **dynamics** of the distant system need to be perceived at the client side. If the pace at which the information is acquired by the server and delivered to the client application is not adequate, the user might get a biased or wrong perception of the actual behavior of the distant system.
In multimedia applications a buffer is traditionally used at the client side to smooth the information playback. Such buffer adds delay to the transmission and therefore cannot be considered if it depreciates the user perception.

The **semantic content** has to be rich enough to enable the perception of the state and the conditions of operation at the client side. There are various options to provide this information. Video image, Virtual-Reality representation or data history can be used for that purpose. For a given type of representation, the more qualitative information is sent, the better the state can be perceived. For instance, a good quality picture, bigger in size, is more informative than a low quality image, smaller in size.

The above three key aspects define the quality of service for RTI2. Since there is no direct RTI2 QoS sensor, metrics needs to be defined using available information. The available information that can be measured or estimated at the client side are the achieved block size and the achieved block pace and, at the server side, the achieved E2E round trip time. The ratios between the measured block values and the initial block characteristics define the metrics: t for the block pace ratio, e for the block size ratio and *d* for the E2E round trip time (Fig. 4).

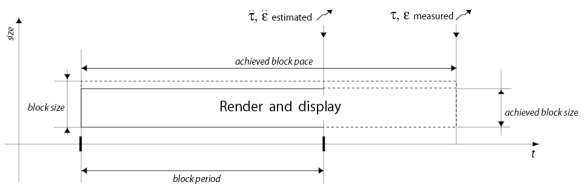

Fig. 4 Metrics definition and measurement

The **block pace ratio** τ is the ratio between the block processing time and the block period. This metric measures the client application ability to handle the received stream of information. This metric is required to ensure that the server transmission parameters are in adequation not only to the network transmission but also to the client characteristics. For a given bandwidth the server application can generate different block sizes and block paces. For example a bandwidth of 100 Kb/sec corresponds to 10Kb blocks sent 10 times per second, but also corresponds to 20 Kb blocks sent 5 times per second. If the client computer such as a PDA were not able to decode and play more than 5 images per second, half of the image would be wasted with the first set of values.

The **block size ratio** ε is the ratio between the successfully rendered block size and the original block size. The successfully rendered block size corresponds to the portion of the block data that was effectively processed and perceived by the user. Blocks can be lost along the E2E transmission path either during the network transmission or at the client side if the client application cannot handle the incoming flow of information. The underlying network protocol can guaranty whether or not the packets are delivered. In the former case, such as in TCP, it can be assumed that the block size ratio is always 1 provided that we wait long enough to receive the complete block and that the client application has enough resource to process the received block. In the later case, without guaranteed delivery, such as in UDP, information can be lost during the transmission. This metric informs the source about the amount of information that has been perceived by the user. An information loss can either be due to a transmission problem or due to a lack of resource in the client application.

The **block round trip time** ***d*** represents the time taken by a block to be successfully sent and acknowledged. The level of interaction can be directly measured by the block round trip time. This measurement takes into account not only the time for the block to transit from the server to the client but also takes into account the processing time at both the client and the server applications. These metrics should be as small as possible to ensure a valuable user experience. Many causes can increase the block round trip time, some are controllable such as the buffers at the server and the client side, like others, the intrinsic Internet connection round trip time cannot be controlled. Block acknowledgements measured at the application level are different than network packets acknowledgements measured at the network layer level.

There are two possible instants to return the measured metrics, the first instant is at the end of the block period and the other one is at the end of the block processing. The former solution presents the advantage of being timely paced and ensures that the next block will be process without further delay. In this case if the time taken to render and display the current block is greater than the block period. If the block rendering and display time is smaller than the block period, $\varepsilon = 1$ provided that the block was received without loss and $\tau < 1$. This former solution carry the disadvantages of requiring metrics estimation for τ and ε if the block processing time (ie the achieved block pace) is greater than the block period and also the truncation of the perceived information. The alternate solution returns the measured metrics but with the drawback of adding a delay to the block acknowledgement. The second method requires less resource at the client side.

4. EXPERIMENTAL RESULTS

Figure 5 graphs the metrics τ function of the E2E bandwidth (y-axis) and function of the block period (x-axis). The two computers used to produce these measurements were connected using a wireless 802.11b link. The pace at which the server sent blocks ranges from 1 block per second (period =1 s) to 20 blocks per second (period = 0.05 s). Along the y-axis the block sizes were chosen to generate an E2E bandwidth ranging from 100 Bytes to 100 kBytes per second. The z-axis represents the metric τ obtained by comparing the block period to the time taken to process and render the given block. When the client application processes the received block at the same space at which it is generated, $\tau = 1$. A value for τ smaller than 1 indicates that the client application is able to handle more information that the server is currently delivering. The server can increase the amount of transmitted data by either by augmenting the block size or by increasing the block pace, or both.

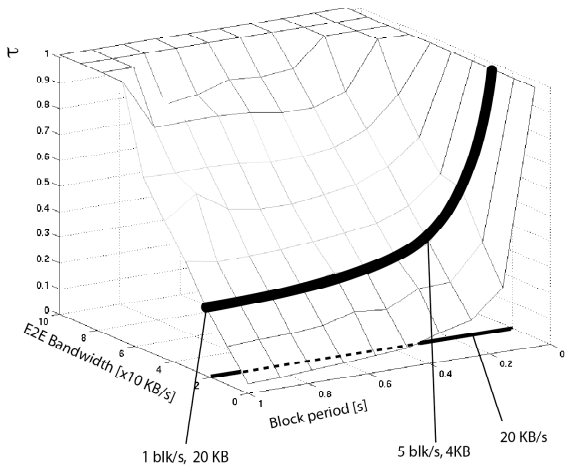

Fig. 5 End-to-end metric τ for transmission over a wireless link.

In this example, when the time taken to process and render the received block is longer than the block period (ie when $\tau > 1$) the client application interrupts the current block processing and start handling the next block in order to limit the E2E round trip time *d*. The result of this behavior is that t has a maximum value of 1, while e is equal or smaller than 1. In this case, the metric e needs to be estimated. An alternative is to pursue the current block processing, resulting in $t > 1$ and $e = 1$. The metrics are therefore returned with an increased delay that is reflected in an increased value of the E2E round trip time *d*.

Those real measurements show that for the same E2E bandwidth, along the x-axis, the value of t varies. This highlights that the bandwidth measurement is not sufficient and strengthens the need for the introduction of the proposed metrics in order to reflect the client ability to handle the received information.

5. CONCLUSIONS

In this paper the metrics needed to measure the quality of service for real-time interaction over the Internet has been introduced. The metrics estimation are made at the application level in order to take into account the whole transmission path from the information capture at the server side to the information rendering at the client side. The proposed metrics are derived from the three key aspects of RTI2 (the level of interaction, the perception dynamic and the semantic content) that defines an efficient user experience. These keys aspects are represented by three quantities - the information transmission delay, the information delivery pace and the information size – that can be measured. Block, the basic unit of information, and component abstractions are introduced to permit the achieved QoS estimation by the metrics.

Experimental measurements strengthens the ability of the proposed metrics to reflect the client ability to handle the received information.

The proposed metrics will permit the development of an adaptation scheme that defines a solution based on the user defined QoS settings and on the metrics measurements.

6. REFERENCES

[1] Gillet D., C. Salzmann and P. Huguenin (2001). "Distributed Architecture for Teleoperation over the Internet". Lecture Notes in Control and Information Sciences 258: Nonlinear Control in the year 2000, Springer-Verlag, 399-407, London.

[2] Floyd S. and K. R. Fall (1999). "Promoting the use of end-to-end congestion control in the Internet". EEE/ACM Transactions on Networking, 7, 4, pp. 458-472.

[3] Feng W.and J. Rexford (1997). "A Comparison of Bandwidth Smoothing Techniques for the Transmission of Prerecorded Compressed Video", Proceedings of the IEEE INFOCOM conference, pp. 58-66.

[4] Aktan B., Bohus C.A., Crowl L.A. and M.H. Shor (1996). "Distant learning Applied to Control Engineering", IEEE Transaction on Education, 39, pp. 320-326.