Analysis of Synchronization Issues for Live Video-Context Transmission Service

Houssein Wehbe, Ahmed Bouabdallah, Bruno Stevant, Usama Mir

Institut Mines-Telecom; Telecom Bretagne
Université européenne de Bretagne
Rennes - France
firstname.secondname@telecom-bretagne.eu

Abstract — *A promising feature brought along the progressive deployment of LTE concerns the increase of the uplink bandwidth. We exploit this new capability by focusing on an innovative usage defined by the simultaneous transmission of live video and contextual data caught through end user devices like smartphones towards websites or distant spectators. The contextual information may indeed be precious to the spectator for several reasons. On one side, this information cannot be deduced from the video images currently displayed. On the other side, being closely related to the live event, context can be fruitfully exploited by the spectator to complete her understanding of what she is presently watching and to possibly interact with the filming person to influence the rest of the capture. The main property of this new feature can be expressed as a synchronization constraint between the video and the contextual data. Ensuring this property is challenging due to the presence of variable delays in the end-to-end path. We focus on particular contextual data (generated by sensors embedded in Android based smartphones) the specificities of which are analyzed in order to derive a general synchronization solution. We finally propose some optimizations taking in account the characteristics of the devices used to display the contextual data.*

Keywords—*Live video; Context; Real-time transmission; Synchronization.*

1. Introduction

Due to the modern day progressive deployment of LTE (Long Term Evolution) the mobile users will soon have accessibility to large bandwidths, with significant improvements in quality of experience (QoE) when receiving multimedia streams [1]. Another promising feature of LTE concerns with the increment of uplink bandwidth. This new capability will foster innovative usages, some of which are built around a basic enabler defined by the direct (or live) transmission towards websites or distant spectators, of live video caught through smartphones. These new generation devices are natively equipped with various chips and sensors providing high processing power and a large variety of information in real time. This information describing different parts of the actual state/environment of the associated device is usually known as the current context and may constitute a precious knowledge when articulated to a communication session [2]. The context may include users' personal and environmental characteristics such as its location, acceleration, camera orientation, temperature, etc [3].

In this paper, we consider the above enabler transmitting together with live video, some contextual data generated from the filming device. This data will be displayed to the spectator through various dedicated GUI (Graphical User Interface) embedded in a context player linked up to the video player. The spectator may therefore advantageously exploit during the live event, the contextual information of the cameraman to complete her knowledge of specific parts of the event and eventually interact with the sender. As a matter of fact the interest in adding to a video stream, data characterizing the context of the sender, rests on helping the spectator to improve her remote involvement in a live event. The brought information may indeed be precious to the spectator because it first cannot be deduced from the video currently displayed and secondly being closely related to the live event, it can be fruitfully exploited by the spectator to complete her understanding of what she is watching at present.

To be pertinent and useful, it is clear that the contextual information must be temporally in phase with the associated video simultaneously received. Such a soundness constraint of this new feature can in fact be expressed as a synchronization requirement between the video and context. This property together with other criteria like video continuity, low initialization delay and low play-out delay (i.e., "live" display of the video), is one of the key issues contributing to ensure the end-user QoE.

We focus in this work on the analysis of the synchronization criterion between video and contextual data, called hereafter "video-context synchronization". It refers to the fact that the relationship between context and video at presentation time, on the receiver play-out device, must match the relationship between context and video at capture time, on the sender device. The simplest way to solve the problem consists in delivering video and context bundled together. Such solution however precludes for example any differentiated treatment between the two media, during transport. It prevents moreover a receiver to get only the video if so desired. These drawbacks disappear when data is transmitted through the network in separate RTP (Real-time Transport protocol) streams [4], but in this case each one might experience different network delays. This increases the probability for receiving and playing one flow ahead of the other which degrades the end-user QoE.

Authorized licensed use limited to: the Leddy Library at the University of Windsor. Downloaded on June 26,2023 at 18:42:29 UTC from IEEE Xplore. Restrictions apply.

Synchronization is one the fundamental issue of computer science raised from its first beginning with the scheduling of concurrent processes of operating systems. The progressive advent of network based computations extended this problematic to the ability to recreate on the receiving side of a communication, the temporal organization of the events occurring at the time of their sending. Even if this problem received in the past a lot of attention [5 - 8], it still constitutes an active area of research confirming by the way, that the "last word" has not yet been said. Beside the lip sync [9] which constituted during a good while the cornerstone use case, new situations regularly appear which instantiate under a new and original point of view the network synchronization problem leading to unexpected developments [10 - 15]. Video-Context synchronization is such a new issue feeding in an interesting way this problematic. It is worth mentioning here that, at our knowledge, it has never been identified before.

The rest of the paper is organized as follows. Section 2 describes use cases highlighting the importance of synchronizing the video with contextual data. Section 3 presents the problem statement and the part of state of the art close to our problematic. Our main solution is explained in Section 4. Section 5 and 6 are devoted to optimizations exploiting some receiver device features. Finally, we conclude our work with some future perspectives in Section 7.

2. USE CASES

In this section, we present two use cases showing the benefits of video-context synchronization in a real-time multimedia service.

Use case 1: We consider one person filming an event with her smartphone and sending both the video and some contextual data in real time to a distant spectator. Figure 1 illustrates an example when this person is driving and the receiver can see her location and velocity vector in a map next to the video player. At instant *t*, the displayed contextual data should match the sender environment when she caught the video image currently displayed. For instance, when the cameraman is crossing by the museum, the spectator should see the museum appearing in the video when the location of the cameraman is shown on the map near the museum.

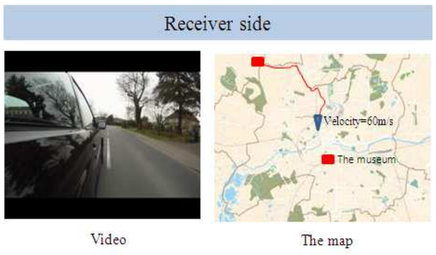

Figure 1: An example showing the relationship between video and contextual data

Contextual data is the appropriate mean to allow the spectator to anticipate the future position of the cameraman and then to interact with her to influence the way the rest of the event will be captured. For example, if the spectator wants the cameraman to slow down to have a better view of the museum, knowing her current position and speed enable the spectator to send in the right time a request before the cameraman actually passes by. Such service is very interesting for the end-user, but it is clear that its soundness is closely related to the synchronization between video and context.

Use case 2: We consider a musical event filmed by six persons with their smartphones. They transmit in real-time the video and some contextual data to a website. At In receiver side, the contextual data such as the location of the filming persons and the smartphones orientation are used to depict a map of the event which can be displayed as indicated in Figure 2. This enables to select the video to be watched based on the characteristics of the environment around the cameraman.

Again, without video-context synchronization, the QoE of a spectator can be heavily impacted. Assume for example, that at instant *t*, cameraman 1 films the river but the map indicates that this person films the musical concert. Spectator selecting the video of cameraman 1 (in order to watch the musical event) will be disappointed and he has to go through all the available videos to find the desired one.

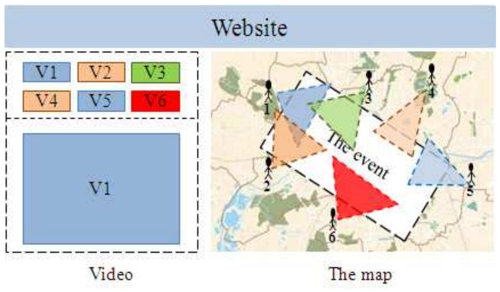

Figure 2: Different persons filming the same event

3. SYNCHRONIZATION IN AUDIO-VIDEO SYSTEMS

3.1. The audio-video synchronization

Synchronizing a video stream with an audio one can be seen as the paradigm of inter-stream synchronization [8]; it will thus help us to introduce the main concepts we will need in the rest of the paper. A general architecture of these existing systems is presented in Figure 3 [16][17].

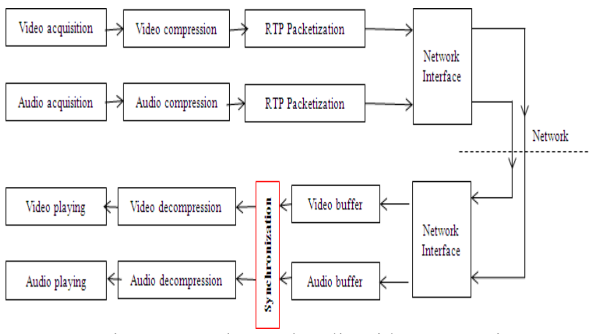

Figure 3: End to end audio-video processing

During the data acquisition step, analog signals are captured and converted into digital format before to be passed

Authorized licensed use limited to: the Leddy Library at the University of Windsor. Downloaded on June 26,2023 at 18:42:29 UTC from IEEE Xplore. Restrictions apply.

to the compression. Then, the data is composed into RTP packets by the packetization step and is sent out to the network via network interface. At the receiver side, the data is first saved in a buffer (e.g., the video buffer) and later decompressed (via data decompression module) in order to be displayed on spectator's screen.

In general, synchronization between audio and video is ensured if the audio and video packets which have been generated at the same instant in sender side, are simultaneously displayed on the receiver device. The possible temporal gap in the alignment of the two streams at the receiver defines the skew between the two media. This notion allows to assess the desynchronization phenomenon appearing when the skew is not null (one stream plays ahead of the other), as shown in Figure 4. Its origin is generally due to the presence of routers and intermediate servers within the end-to-end path.

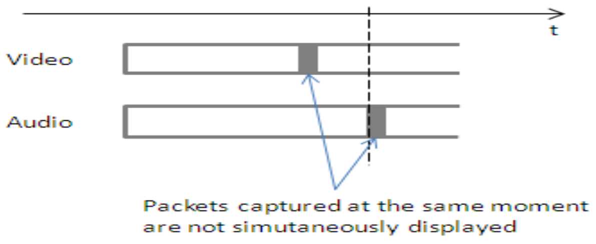
t
Video
Audio
Packets captured at the same moment are not simutaneously displayed

Figure 4: Example of a desynchronization phenomenon

[19] provides a detailed study of the end user capability to detect harmful impacts of desynchronization on QoE. First, video ahead of audio appears as less of a problem than the reverse and secondly an absolute skew smaller (resp. greater) than 160 ms (resp. 320 ms) is harmless (resp. harmful) for QoE. The author identifies a double temporal area [-160,-80] and [80,160] called transient, in which the impact of the skew heavily depends on the experimental conditions. Mobile environment specificities (size of the image, distance of the user from the screen, video frame rate) don't bring such results into question [20]. This set of temporal skew thresholds (in sync, transient, out of sync) [19] constitutes a fundamental parameter of inter-stream synchronization. It is however clear that the particular thresholds values are closely related to the nature of the various media involved in the RTP streams.

In real implementations, the transient area has to be spreaded out between the two other ones to allow simple algorithmic solutions [21][22]. During reception, when the in sync threshold is satisfied, data is directly sent to decompression (cf. Figure 3) since it is considered as synchronous. Otherwise, the receiver applies the following process [22]. If the video content has been captured before the audio content, for the spectator point of view, the video appears to be delayed. The video playing module (cf. Figure 3) skips out some video packets until re-synchronization of audio-video occurs. On the other side, if the audio is delayed, the newly received video packets are stored in a buffer and the video playing module continues to play the old packets repeatedly, until they are synchronized. This asymmetric treatment between audio and video is due to the fact that lip sync attributes highest priority to audio. This is usually expressed by saying that the audio stream is the master one.

Lip sync necessitates restoring at the receiver side a temporal snapshot of the sender situation, to be able to detect when the various thresholds are satisfied. The trick consists in marking during the RTP packetization (cf. Figure 3), each packet with a timestamp representing the capturing moment of the associated content. Due to the use of different clocks when generating the timestamps of audio and video, a common reference clock called the wallclock, which can be instantiated for example by NTP (Network Time Protocol) [18], is required to correlate the timestamps to a common base time. The relationships between the sender local clocks and NTP are sent to the receiver in RTCP reports [4].

Such a solution solves the synchronization problem very simply. The main point rests on restoring at the receiver side a temporal snapshot of the sender situation. It is clear that this principle is independent of the nature of the data, the streams of which have to be synchronized.

3.2. Contextual data

Before applying the previous ideas to the video-context synchronization, we recall the nature of contextual data generated through smartphones, which is rather different from that of audio or video.

Let's take the case of a first application A1 deployed in an Android based smartphone. A1 collects and pushes toward a second application A2, contextual data generated by embedded sensors. Although not necessary, we suppose however that A2 is deployed in an external entity to keep coherence with our initial problematic. To de-correlate A2 needs from the sensors generation capabilities, Android 4.1 (Jelly Bean) defines five different policies allowing A1 to monitor the information provided by the sensors, by fixing a specific filtering rate which can be different from the sampling rate of the sensors [23]. The underlying idea is that each policy should be tailored to the capabilities of the consuming application namely A2, trying to master by the way processor load and energy consumption. The interval between two sensor events selected by A1 can therefore be user defined, equal to 200, 60, 20 microseconds or possibly null (the associated filtering rate is respectively user defined, equal to 5, 16, 50Hz or possibly infinite). The first four cases determine A2 as a kind of application regularly consuming contextual data. The last one is well adapted to reactive applications interested in any generated contextual data. It is however worth quoting the previous reference, to point here that the situation could be a little bit more complex : "*The delay that you specify is only a suggested delay. The Android system and other applications can alter this delay. As a best practice, you should specify the largest delay that you can because the system typically uses a smaller delay than the one you specify*" [23]. In the same way, [24] provides an interesting experimental study mentioning similar average rates (~10Hz or less) for contextual data such as acceleration (48Hz), magnetic field (60Hz), gyroscope (870Hz), etc. It is clear that this specific point should deserve much more

Authorized licensed use limited to: the Leddy Library at the University of Windsor. Downloaded on June 26,2023 at 18:42:29 UTC from IEEE Xplore. Restrictions apply.

attention through comprehensive experimental studies. It finally appears that Android's built-in capabilities provide interesting means to precisely regulate the rate of the generated data pushed toward A2.

A sensor event when generated, is transferred to the Android operating system which encapsulates it in a structured 4-tuple containing the raw sensor data, the type of the associated sensor, the accuracy of the data and the timestamp (in nanosecond) at which the event happened [23]. It can be easily seen that the size of such a structure is roughly of the order of few dozens of bytes which is close of the typical size of an audio frame [25].

We can finally draw several conclusions which can be seen as the requirements of our problematic. First, an Android-based application pushing contextual data generated by embedded sensors can specify an upper bound (built-in or user-defined) on the pushing rate or accept a variable one. The order of magnitude of the contextual data generation rate can however be much lower from the one usually associated to video (90kHz [25]) or audio (8kHz, 11.025kHz, 16kHz, 22.05kHz, 24kHz, 32kHz, 44.1kHz, 48kHz [25]). Concerning the elementary amount of information, context is nevertheless very close to audio.

3.3. RTP capabilities

RTP can be used to transport other data than audio or video. We can quote in this sense several dedicated payloads which have been specified :

- real-time pointers [26]
- text conversation [27]
- DTMF digits, telephony tones and telephony signals [28]

One of their common points concerns the used sampling rate which is always constant and respectively equal to 90kHz, 1kHz, 8kHz.

We however have an interesting illustration of the RTP capabilities when it has to cope with variable rates produced data. [29] defines a payload dedicated to the Speex codex, one characteristic of which concerns its variable bit-rate. Speex introduced ten years ago, constitutes an interesting attempt to define a free-patent codec dedicated to VoIP [30]. It has been deprecated since the recent and official advent (September 2012) of the Opus codec [31] sharing for our concern, the same characteristic as Speex. We however didn't find for it the definition of a dedicated RTP payload. That is why we focus on the Speex case.

Speex is a 20 ms frame-based codec encoding inside each frame both the sampling rate and the bit-rate associated to the current frame. For the same bit stream, these parameters can therefore vary, inside specified ranges, from frame to frame leading to dynamic switching between variable bit-rates. The used encoding technique indifferently authorizes narrowband, wideband or ultra-wideband determining as sampling rate respectively 8kHZ, 16kHz and 32kHz.

4. SOLVING THE VIDEO-CONTEXT SYNCHRONIZATION PROBLEM

In a first work [32], we proposed a solution restricted to one video stream associated with one constant pushing rate context stream. In a second work [33], we extended our analysis to multiple context streams and focused on the particular case of a variable pushing rate for the generated contextual data. We now define in this paper a general framework which may cope indifferently with variable or upper bounded pushing rates.

Let's come back to the use case initiated in § 3.2 where A1 collects multiple contextual data generated by sensors embedded in a smartphone. These ones are pushed in separate RTP streams toward a second application A2 deployed in an external entity, together with a live video stream filmed by the smartphone. We suggest adapting the lip sync method to synchronize all these various streams at the receiver [9][16][17]. We define the video stream as the master one. We suppose given an RTP payload dedicated to the transport of context as suggested in [33]. The tricky point concerns the current pushing rate of contextual data. On one hand it can be user defined, or equal to 5, 16, 50Hz, on the other hand it can vary in an unpredictable way. We suggest extending the Speex RTP payload principle by embedding the current pushing rate together with the timestamp of the associated context information structure inside the payload. In the Speex case however, the purpose of the embedded information consists in adjusting the right decoding technique on the receiver side. In our case, the interest of this information is closely related to synchronization purposes. It provides indeed an elegant way to map the timestamp of the associated RTP packet to the wallclock on the receiver side. In this way we can use a variation of the lip sync to carry out the desired synchronization between the video and the context streams.

We propose in Figure 5 below a precise definition of the algorithm which should be applied by the receiver, whenever a new context packet is received. Let V, |V| and C be respectively the next video packet to display, its associated display duration and any context packet. Suppose Tc (resp. Tv) is the wallclock associated to the timestamp of C (resp. V). The video being the master stream, we identify the following three main cases :

- $Tc < Tv$: the context packet is out of date, it can therefore be destroyed.

- $Tv \leq Tc < Tv + |V|$: the two packets are synchronized, they must be displayed.

- $Tv + |V| \leq Tc$: the context is ahead the video, it must be bufferised until the associated video packet arrives, this one has to be resynchronized with the context.

To ensure a smooth display of the contextual packets, we suggest applying the above algorithm also when the display of a context or a video packet arrives to its end. At this moment, the receiver consults the buffer and looks for the next context packet which can be identified by using the RTP sequence number. If this packet, e.g., C, is available, the receiver selects it and compares its timestamp Tc with that of the current video packet Tv as described in the previous algorithm, and

Authorized licensed use limited to: the Leddy Library at the University of Windsor. Downloaded on June 26,2023 at 18:42:29 UTC from IEEE Xplore. Restrictions apply.

decides then either to play this packet or to restock it until data is resynchronized. Otherwise, the receiver detects a discontinuity of display because the desired context packet is either lost or not yet received. We suggest in this case applying the optimization algorithm presented in section 5 below to guarantee a smooth display of the contextual data.

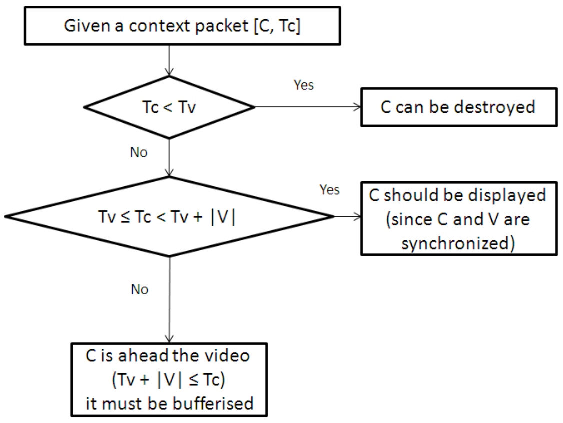
Given a context packet [C, Tc]
Tc < Tv
Yes
C can be destroyed
No
Tv ≤ Tc < Tv + |V|
Yes
C should be displayed (since C and V are synchronized)
No
C is ahead the video (Tv + |V| ≤ Tc) it must be bufferised

Figure 5. General synchronization algorithm

Continuing above, Figure 6 shows an example of the three possible cases that can appear on receiver side. Let T and |V| be respectively the timestamp of the first video packet (Vi-1) and its duration. We assume that the sender has generated three video packets and four context packets as drawn in the figure.

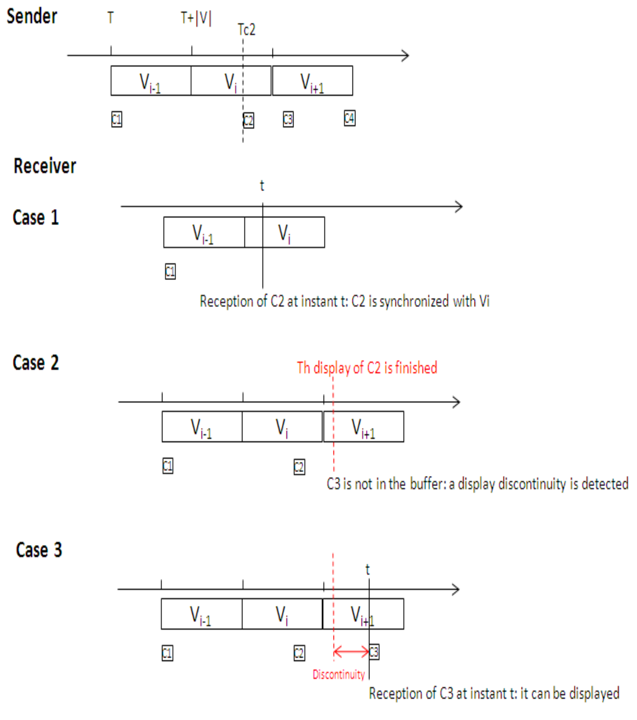
Sender
T
T+|V|
Tc2
V_{i-1}
V_i
V_{i+1}
C1
C2
C3
C4
Receiver
Case 1
t
Reception of C2 at instant t: C2 is synchronized with Vi
Case 2
Th display of C2 is finished
C3 is not in the buffer: a display discontinuity is detected
Case 3
t
Discontinuity
Reception of C3 at instant t: it can be displayed

Figure 6. Some possible cases

It is to be noted that the relationship between the video and context packets at the sender should be retrieved on receiver side to ensure the synchronization. We consider in case 1, that the context packet C2 is received at the right time. The receiver compares its timestamp with the one of Vi, as described in the previous algorithm, and displays it on the screen because C2 has been generated before the capture of Vi arrives to its end. Furthermore, we consider in case 2 that when the display of C2 is finished, the context packet C3 is not yet received. In this case, the receiver detects a discontinuity of display and applies then the algorithm we will present in section 5. But when the context packet C3 arrives to the receiver (case 3), it will be displayed because the video packet Vi+1 is currently under display on the screen.

In the case of a video packet received before the associated context packet, the receiver plays the video (because the video always has the priority) and informs the spectator with a dedicated error message. It is also possible to exploit the nature of contextual data to express it in a more intuitive and natural way. For example, in use case 1 presented in Section 2, when the desynchronization occurs, the system can automatically zoom out from the detailed map to show a more general overview of location to the spectators.

Our solution works well in the case of a single context stream attached to a video stream. Declining this solution when multiple contexts (such as velocity and location) are attached to a single video stream, simply consists in separately synchronizing with our algorithm, each context stream with the video stream.

5. Play-out devices features based optimization

This section presents an optimization taking in account the display of the contextual data. The receiver is equipped with context play-out device characterized by its accuracy of display, called hereafter the *accuracy*. For example, the map (cf. Figure 1) displays new position when the real location varies at least 20m such that *accuracy*=20. After receiving and decompressing a context packet C_i, the receiver obtains a value called $V(C_i)$. It will be compared with the value of currently displayed context packet $V(C_{i-1})$. If $|V(C_i) - V(C_{i-1})| > accuracy$, then $V(C_i)$ is displayed on the screen. Otherwise, there is no need to display $V(C_i)$ since the spectator cannot see the difference (on map) in comparison with $V(C_{i-1})$. We suggest exploiting this property to improve the algorithm introduced in the previous section.

To take in account the above characteristics of a play-out device, we propose to apply the following technique after detecting a discontinuity of display or a desynchronization phenomenon. At this moment, the receiver estimates the value of the missing context packet $V(C_i)$ and compares it with that of the current context packet $V(C_{i-1})$. If the difference is smaller than *accuracy*, the receiver continues to display C_{i-1} instead of C_i because the spectator cannot see the difference between these packets. Otherwise, the receiver shows an error message and can either display the estimated value $V(C_i)$ or exploit the nature of contextual data, for example by zooming out from the detailed map to show a more general overview of location to the spectators.

Authorized licensed use limited to: the Leddy Library at the University of Windsor. Downloaded on June 26,2023 at 18:42:29 UTC from IEEE Xplore. Restrictions apply.

The challenge of this technique is how to know the value of packet that is not already received, i.e., $V(C_i)$. We can extrapolate it using the previous values of the received context. For example, if d is the difference between the two previous values :

$$d = (V(C_{i-1}) - V(C_{i-2}))$$

then we approximate the value of C_i as follows :

$$V(C_i) = V(C_{i-1}) + d$$

This estimation enables to get a true value mainly in steady state, e.g., when the sender is driving with constant speed. As a result, this algorithm enriches the lip sync solution by taking in account the context play-out device characteristics.

6. Resources optimization through receiver feedbacks

Due to the nature of the contextual data and the characteristics of its play-out device, we consider that there are new possibilities to optimize the use of network and end-point resources. Indeed, some contextual data, e.g., the location and the temperature, does not change continuously. Sending permanently a packet of this data increases the probability for receiving successive packets having the same information. Due to the characteristics of the context play-out device, these packets can be considered as redundant. As a result, the sender transmission resources are not properly utilized as it does not take into account the receiver requirements. To deal with this problem, we propose to add the following algorithms.

The receiver informs the sender about the current *accuracy* value configured for her context display device. RTCP packet such as the Application-specific message (APP) can be used to ensure the transmission since it enables to design application-specific extensions to the RTCP protocol [4]. This message is sent initially and when the play-out device characteristics are changed, for example when the spectator zooms in the map.

The sender adapts the emission rate of the contextual data with respect to the receiver requirements. After data acquisition phase, it compares the acquired data with the content of the last sent packet. Based on the display accuracy of the receiver, the sender decides if the new data will be sent or discarded. Data is sent only if it will be displayed in receiver side. Assume, for example, that the map of the receiver displays new position each 20m. The context packets indicating that the sender location is changed for distance smaller than 20m, will be discarded but all other packets are sent.

The above algorithm can be applied in a scalable live video streaming system. It uses the RTP/RTCP messages to ensure the interactivity between the sender and the receiver. These transport protocols are scalable and there is a very small probability that the RTCP messages (the feedbacks) generate a congestion phenomenon in sender side since they usually represent at most 5% of the RTP data [4]. The sender resources are properly used because it does not emit unnecessary packets. This enables to reduce the number of transmitted packets and therefore the congestion probability.

7. Conclusion

Recently, the mobile devices have become an innovative way to produce new communication services involving and exploiting the user context information. In this paper, we considered a service allowing the transmission of live video together with contextual data from a mobile device to a distant spectator. We focused on the video-context synchronization problem both with constant and variable pushing rates and proposed optimization techniques by taking in account the characteristics of end-user devices used to capture and exploit the contextual data. Our solution detects a false desynchronization phenomenon and enables to properly use the end-point resources as new packets are only sent when required.

A future work will consist in adapting the application Zewall [34] to evaluate the correctness of our synchronization algorithm through real experimentations. These ones should be by the way, exploited to explore the display characteristics of contextual data with respect to its type and nature (temperature, light, pressure, orientation, etc).

Such experimentations will require the definition and the development of an RTP payload dedicated to context. Another associated issue will will lead us to compare the pro and cons of other transport protocols [35][36] relatively to RTP.

Acknowledgments

This work is part of the collaborative project Zewall funded by the French state and the Brittany region.

The authors thank Marco Zavatta for pointing out incorrect values of the various pushing rates.

References

[1] Liang Zhang, T. Okamawari, T. Fujii, "*Performance Evaluation of End-to-End Communication Quality of LTE*", 75th IEEE Vehicular Technology Conference (VTC Spring), 6-9 May 2012, pp.1-5.

[2] F. Toutain, A. Bouabdallah, R. Zemek, C. Daloz, "*Interpersonal context-aware communication services*", IEEE Communications Magazine, January 2011, vol. 49, n° 1, pp. 68-74, 2011.

[3] A. Corradi, M. Fanelli, L. Foschini, "*Adaptive context data distribution with guaranteed quality for mobile environments*", 5th IEEE International Symposium on Wireless Pervasive Computing (ISWPC), 5-7 May 2010, pp.373-380.

[4] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "*RFC 3550, RTP: A Transport Protocol for Real-Time Applications*", the Internet Society, July 2003.

[5] G. Blakowski, R. Steinmetz, "*A Media Synchronization Survey : Reference Model, Specification, and Case Studies*", IEEE Journal On Selected Areas In Communications, January 1996, vol. 14, n° 1, pp. 5-35, 1996.

[6] Y. Ishibashi, S. Tasaka, "*A Comparative Survey of Synchronization Algorithms for Continuous Media in Network Environments*", Proceedings of the 25th Annual IEEE Conference on Local Computer Networks (LCN 2000), 8-10 November 2000, USA, pp. 337 - 348.

[7] N. Laoutaris, I. Stavrakakis, "*Intrastream Synchronization for Continuous Media Streams : A Survey of Playout Schedulers*", IEEE Networks, May/June 2002, pp. 30-40.

[8] F. Boronat, J.Lloret, M. Garcia, "*Multimedia group and inter-stream synchronization techniques : A comparative study*", Information Systems, n°34, pp. 108-131, 2009.

[9] I. Kouvelas, V. Hardman, A. Watson, "*Lip Synchronisation for use over the Internet : Analysis and Implementation*", Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '96), 18-22 November 1996, London, England, pp.893-898.

Authorized licensed use limited to: the Leddy Library at the University of Windsor. Downloaded on June 26,2023 at 18:42:29 UTC from IEEE Xplore. Restrictions apply.

[10] Y. Ishibashi, S. Tasaka, "*A Distributed Control Scheme for Causality and Media Synchronization in Networked Multimedia Games*", Proceedings of the IEEE International Conference on Communications (ICC 2001), 11-14 June 2001, Helsinki, Finland, pp.952 – 958.

[11] O. Wongwirat, S. Ohara, "*Haptic Media Synchronization for Remote Surgery through Simulation*", IEEE Multimedia, July-Sept. 2006, vol. 62, n° 3, pp. 62-69.

[12] J-K. Yun, J-H. Jang, K-D. Moon, "*Five Sense Media Playback Technology using Multiple Devices Synchronization*", Proceedings of the 5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT), 30 Nov. – 2 Dec. 2010, Seoul, pp.73 – 75.

[13] C. Perkins, T. Schierl, "*RFC 6051 : Rapid Synchronisation of RTP Flows*", the Internet Society, November 2010.

[14] S. Hoshino, Y. Ishibashi, N. Fukushima, S. Sugawara, "*QoE Assessment in Olfactory and Haptic Media Transmission : Influence of Inter-Stream Synchronization Error*", Proceedings of the IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR 2011), 10 – 12 May 2011, Naples, USA, pp. 1 – 6.

[15] F. Boronat, M. Montagud, V. Vidal, "*Master Selection Policies for Inter-Destination Multimedia Synchronization in Distributed Application*", Proceedings of the 19th Annual IEEE International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2011), 25 – 27 July 2011, Singapore, pp. 269 – 277.

[16] C. Perkins, "*RTP : Audio and Video for the Internet*", Ed. Addison-Wesley, 2003.

[17] S. Firestone, T. Ramalingam, S. Fry, "*Voice and Video Conferencing Fundamentals*", Ed. Cisco Press, 2007.

[18] D. Mills, J. Martin, J. Burbank, W. Kasch, "*RFC 5905, Network Time Protocol Version 4: Protocol and Algorithms Specification*", the Internet Society, June 2010.

[19] R. Steinmetz, "*Human Perception of Jitter and Media Synchronization*", IEEE Journal On Selected Areas In Communications, January 1996, vol. 14, n° 1, pp. 61-72, 1996.

[20] I.D. Curcio, M. Lundan, "*Human Perception of Lip Synchronization in Mobile Environment*", Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, (WoWMoM 2007), 18-21 June 2007, Helsinki, Finland, pp. 1 – 7.

[21] F. Boronat Segui, J.C Guerri Cebollada, J. Lloret Mauri, "*An RTP/RTCP based approach for multimedia group and inter-stream synchronization*", Multimedia Tool Applications, n° 40, pp.285-319, 2008.

[22] J. Zhang, Y. Li, Y. Wei, "*Using Timestamp to Realize Audio-Video Synchronization in Real-time Streaming Media Transmission*", Proceedings of the 2008 International Conference on Audio, Language and Image Processing, Shanghai, China, 7-9 July 7-9 2008, pp. 1073-1076.

[23] Android 4.1 Jelly Bean : Sensors Overview → http://developer.android.com/guide/topics/sensors/sensors_overview.html

[24] S. Ubejd and R. Angel. *"Indoor Positioning using Sensor-fusion in Android Devices"*, Student thesis, Kristianstad University Sweden, School of Health and Society. September 2011. http://hkr.diva-portal.org/smash/record.jsf?pid=diva2:475619

[25] H. Schulzrinne, S. Casner, "*RFC 3551, RTP Profile for Audio and Video Conference with Minimal Control*", the Internet Society, July 2003.

[26] M. Civanlar, G. Cash, "*RFC 2862 : RTP Payload Format for Real-Time Pointers*", the Internet Society, June 2000.

[27] G. Hellstrom, P. Jones, "*RFC 4103 : RTP Payload for Text Conversation*", the Internet Society, June 2005.

[28] H. Schulzrinne, T. Taylor, "*RFC 4733 : RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals*", the Internet Society, December 2006.

[29] G. Herlein, J. Valin, A. Heggestad, A. Moizard, "*RFC 5574 : RTP Payload Format for the Speex Codec*", the Internet Society, June 2009.

[30] Speex codec → http://www.speex.org/

[31] J. Valin, K. Vos, T. Terriberry, A. Moizard, "*RFC 6716 : Definition of the Opus Audio Codec*", the Internet Society, September 2012.

[32] H. Wehbe, A. Bouabdallah, B. Stevant. *"Synchronization Mechanisms for Live Video-Context Transmission Service"*, NGMAST. The 6th IEEE International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST2012). Paris, September 2012.

[33] A. Bouabdallah, H. Wehbe, B. Stevant. *"Uplink Transfer of Live Video Synchronized with Multiple Contextual Data"*, Media Synchronization Workshop, Berlin, 11 October 2012, ISBN N° 978-90-5986-410-8.

[34] Zewall project → http://brest2012.zewall.eu

[35] N. Aschenbruck, C. Fuchs, *"STMP – Sensor data Transmission and Management Protocol"*, Proceeedings of the 36th Conference on Local Computer Networks (LCN 2011), 4 – 7 october 2011, Bonn, Germany, pp. 475 – 483.

[36] Z. Shelby, K. Hartke, C. Bormann, B. Franck, *"Constained Application Protocol (CoAP) – draft-ietf-core-coap-12"*, the Internet Society, October 2012.

Authorized licensed use limited to: the Leddy Library at the University of Windsor. Downloaded on June 26,2023 at 18:42:29 UTC from IEEE Xplore. Restrictions apply.