SESSAMO: SESSION MOBILITY FOR VIDEO STREAMING APPLICATIONS

Mohammed BOUTABIA[1], Luis Rojas CARDENAS[2] and Hossam AFIFI[1]

[1]TELECOM SudParis, CNRS SAMOVAR UMR 5157
9, Rue Charles Fourier - 91011 Evry Cedex, France
{mohamed.boutabia,Hossam.afifi}@it-sudparis.eu
[2] Universidad Autonoma Metropolitana
Vicentina DF, Mexico
lmrc@xanum.uam.mx

ABSTRACT

Nowadays, telecom operators are making a remarkable progress in providing a wide offer of broadband access to answer the high demand for high bit rate applications. Nevertheless, user requirements do not stop at providing high rate connection, but exceeds it to ensuring transparent service portability among his equipments. The user would like to choose among his devices those which respond at best his needs and constraints. From small smart phones to large screen devices, the customer enjoys its entertainments or business meetings according to its current situation. Service continuity over different terminals known as session mobility is a challenging operation in terms of handover latency, context transfer and media adaptation. Moreover, this transfer requires synchronization between the involved terminals.
In this paper we present "SESSAMO", a new lightweight session mobility protocol for streaming applications using RTSP. This solution is transparent to the network and does not require any changes in the client or server streaming application. The solution is detailed and a set of measurement results are presented. In addition, we present a new method to renegotiate session parameters following terminal capacities in order to adapt the flow accordingly. Renegotiation proposal is based on the use of SDPng and MPEG-21.

KEYWORDS

MPEG21-DIA, RTSP, SDPng, Session mobility, Video streaming

1. INTRODUCTION

The goal of session mobility is to give the users the possibility of switching from one terminal to another when enjoying the same multimedia session without any interruption. This operation proved to be useful for users who are in mobility. A typical scenario of session mobility is a user who is using his personal data assistant (PDA) to watch his favorite game when he is outside. The PDA is connected to internet via the public land mobile network (PLMN) which provides 3G service. Once at home the user would like to take advantage of his broadband access and his high definition screen to watch the same media without any service initialization or interruption. Session transfer can be either controlled by the device from which the session is transferred (see Figure 1), this mode is called push mode, or by the device to which the session is transferred, this mode is called pull mode.

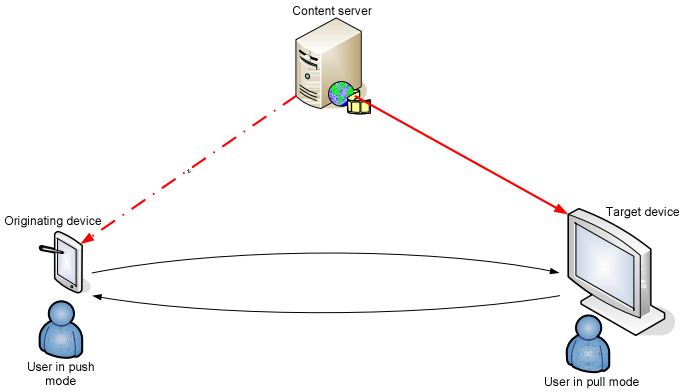

Figure 1: push mode and pull mode in session mobility

1.1 Service continuity and its constraints

Service continuity means that there should be no interruption when the media stream is transferred between the two devices. Therefore, session mobility adds a temporal constraint to the transfer operation. Handover delay is the period of time elapsed from the instant when the user chooses to switch to the new device by triggering the transfer (ex: button push) and the time instant when the stream starts playing in the target device. In other words, handover delay represents the reactiveness of the transfer mechanism. In the ideal case there should be no time difference between the instant when the media disappears from the first device and the instant it appears again in the target device. In practice the handover delay is not null due to the delay in transmitting signaling packets and data packets between the involved entities (i.e. media server, target device and originating device). Nevertheless, handover delay should be minimized in order not to disturb user's quality of experience. Service continuity adds another constraint related to synchronization of the media between the two devices. An accurate session transfer requires that the stream starts in the target device from the instant it was left off in the first device. This issue is a direct consequence of the handover delay. Solving this issue is important in order to preserve the consistency of the service. This means that we must be sure that the user doesn't miss any sequence of the media and minimize any overlapping in the played material (i.e. play a sequence already displayed in the originating device).

1.2 Media adaptation

Another challenge of session mobility is adaptation of the media being transferred. The variety of devices and their capacities makes it obligatory for the media streams played by these devices to be adapted to their capacities and connectivity rate. For instance the media stream played by the high definition screen is not suitable for a mobile device without any adaptation; otherwise the system will be overloaded and the quality will be very poor.

The difficulty of media adaptation resides in how to adapt the media streams contained on servers to the big variety of screens, CPU capacities, network rates, batteries drain…etc. This process involves a number of tasks where signaling procedures are not completely defined. To achieve this task two procedures are distinguished: negotiation and adaptation. Negotiation is initiated by the client and takes place at the beginning of the session or during the session when one of the parameters change. Adaptation is the action taken by the server after the negotiation in order to make the necessary changes on the served streams.

2. RELATED WORK

2.1 Mobile IP

MIP is not suitable for session mobility although it can be a considered as a candidate solution. MIP can achieve session mobility since the new terminal has a new IP address and MIP can redirect data packets from the old address to the new address which is actually a new terminal. Nevertheless, not only the session will be transferred, but every thing that was destined to the origination device will be routed to the new device. This is not the objective of session transfer in video streaming where the transfer concerns only the stream itself.

2.2 Real Time Streaming Protocol

Real Time Streaming Protocol (RTSP) [1] is an application level protocol providing signaling service for real-time data delivery such as audio and video. Management of the session is provided by a set of methods implementing the classical control player actions such as play, record, pause and stop. Other methods are concerned about describing and negotiating session parameters. RTSP does not provide any data delivery by itself, but relies on transport protocols such as Real-time transport protocol (RTP) [2]. RTSP operation is based on the client/server approach where the client sends request to the server, and the server answers the client requests. Figure 2 illustrates RTSP session establishment and termination. As far as session mobility is concerned, RTSP does not provide any mechanism for session transfer. For this reason external mechanisms should take in charge this operation.

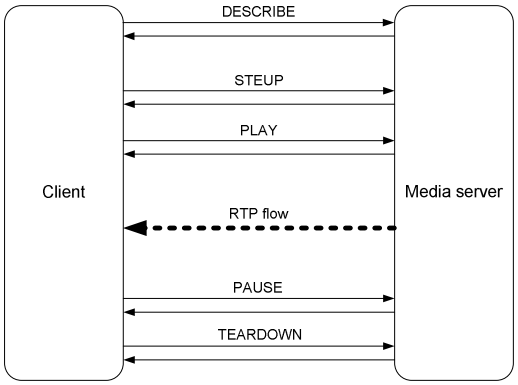

Figure 2: Session establishment and termination in RTSP

2.3 Session Initiation Protocol: Refer method

SIP supports many types of mobility such as terminal, session, personal and service mobility by exchanging a number of messages between the concerned entities. At the opposite of RTSP, SIP provides a built-in method for session transfer between different user agents. REFER [3] is an extension method that allows a client to transfer the session to a third party. Many other works are based on the use of SIP session mobility to provide a framework for session transfer as [4][5].

3. PROPOSED SESSION MOBILITY MECHANISM

As the purpose of our study is providing service continuity for video streaming, we will choose the RTSP as basis of our solution. RTSP is more suitable for this kind of service rather than SIP which is more adequate for reactive applications like telephony and videoconference services.

Therefore, a new mechanism for session mobility should be defined and integrated to the streaming application.

3.1 Session mobility operation

Session mobility allows the user to maintain its session when changing end terminals. The action is started when the user pushes the button to indicate that he/she wants to switch the flow from the first terminal to the second one. A request is immediately sent to the remote terminal. This message contains a description of the current session in an SDP like format. If the new terminal accepts the solicitation, the application sends a command to launch the viewer and the negotiation procedure between the new terminal and the server starts. After negotiation, the target device requests the flow from the server. At the end of this phase, the target terminal begins receiving the flow with requested characteristics; subsequently it sends a message to the first terminal to indicate that the process succeeded. Ending the old session can follow two approaches: make-before-break and break-before-make. Make-before-break approach consists of making sure that the stream is received by the target device before stopping it from the originating one, therefore the acknowledgement is sent only if the data packet is arriving. Whereas break-before-make approach consists simply of acknowledging the transfer request at its reception by target device therefore, the session is stopped in the originating device without any guaranty that the actual session was transferred successfully. Finally, the original RTSP entity sends a TEARDOWN message to the server to stop receiving the flow.

3.2 Protocol description

SESSion And MObility (SESSAMO) protocol is a simple protocol that takes place between the two devices in a peer to peer relationship. It means that both devices implement server and client modules. SESSAMO is based on the exchange of text messages [6] like HTTP and RTSP. Two main kinds of messages are identified: request messages and response messages. The first conveys command information, whereas the second one transports the status of the operation result. The general format of a SESSAMO message is as following:

Header 1 CRLF
Header 2 CRLF
...
Header n CRLF

A header is always composed of two elements: the header identifier and the value or attribute. Inside the message, the header can take any order. On the other hand, there is a reduced set of headers so that total length of the message does not exceed the Maximum Transfer Unit (MTU). Hereafter the list of the headers defined for session transfer:

- “type: session ”: this header determines the type of the message.
- “sequence: sequence number”: this header indicates the sequence number n of the header. It is useful for controlling message loss and message duplication as well as for security.
- “time: time in seconds”: this header specifies the time instant t in seconds at which a session has to start from.
- “service: URL”: this header specifies the URL (Uniform Resource Locator) from where the session can be obtained. This is the location of the media server serving the current streaming.
- “status: code”: this header reports the result of the requested operation. When status code is equal to 1, the operation was successful, whereas 0 means that the operation has failed.

The protocol operation is simple: for each request message there is a response message. Moreover, they have to share the same sequence number. If the sequence number is different, there is a lost or a duplicate message, therefore other measures have to be taken to deal with this inconsistency. If a response message is not acknowledged in a period of RTT seconds (100 ms if RTT is not known), it has to be retransmitted, where the maximum number of retransmission is seven. A typical request message for session transfer is:

- type session CRLF
- sequence 3 CRLF
- time 1040.36 CRLF
- service RTSP://157.159.103.232:8554/ice_age CRLF

And its corresponding response is:

- status 1 CRLF
- sequence 3 CRLF

3.3 Implementation

We use the open source library live555 [7] under linux for RTSP protocol and VLC [8] as a video player in the end devices. Live555 is used as video server and also included as a module during the compilation of VLC player for client support of RTSP. SESSAMO coordinates between the two entities participating in session mobility operation: the originating device (OD) and the target device (TD).SESSAMO is a peer to peer protocol, for this reason any device that supports session mobility has two kinds of programs: SESSAMO server which listens in permanence to requests coming from other terminals, and client part that can transfer the session from the current terminal. Thereby, SESSAMO is operating in push mode. Client program is equipped with a graphical user interface to coordinate the operation between both entities. It includes a set of graphical resources to establish the operating parameters. On the one hand, it controls VLC to accomplish commands such as play, stop, pause…etc. On the other hand, it retrieves status reports about the current video session, such as the last received RTP timestamp. This information is used later in the session transfer operation in order to achieve the session handover accurately. The exchange of information required by the protocol is always started by OD. This occurs when the user pushes the key “transfer of current session” (see Figure 3). This action makes the SESSAMO protocol send a request message to TD. This message contains the necessary information to retrieve the video session. If the TD does not receive a response before RTT seconds, it retransmits the same message; the operation is repeated if necessary but not more than seven times. After seven attempts, the communication with the TD is considered as impossible.

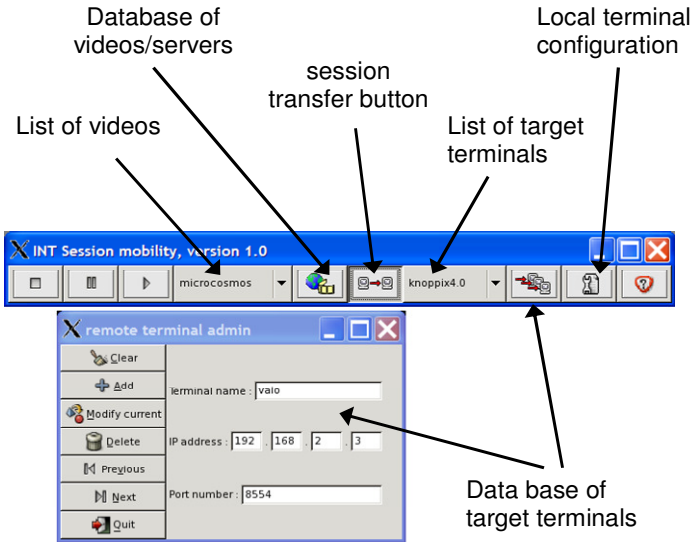

Figure 3: SESSAMO client graphical interface

As far as server program is concerned, it is a daemon running in the background which listens on a special port to the incoming requests from OD and treats them. When the TD receives a request message, it recovers the session description and launches the RTSP player under the following conditions: (1) the video session must be retaken from the point specified by the request message and (2) the video to be negotiated has to be compliant with the TD capabilities.

3.4 Testbed

The goal of this testbed is to validate and test the efficiency of our solution in a real video streaming scenario. The testbed is composed of a PC playing the role of the video server; the TD is represented by a laptop, and Nokia 770 internet tablet is the OD (see testbed snapshot in Figure 4). The internet tablet is connected via WIFI to the LinkSys access point which is linked to the other entities through an Ethernet Hub. The tablet is playing the role of the mobile device and the laptop is the high capability device. SESSAMO client and server are developed in C language. As Nokia internet tablet has a different environment (i.e. ARM processor) we compile SESSAMO program using scratchbox [9] (see Figure 5) which is a cross-compiler provided by Nokia allowing the development and compilation of new applications destined to work on that tablet.

Figure 4: testbed snapshot

Table 1 summarizes the different elements of the testbed and their hardware capabilities.

Table 1: Characteristics of testbed elements

Element	Function	Hardware description
Nokia 770	Originating device	252MHz OMAP, 64 Mo RAM, 802.11b/g
Laptop Dell	Target Device	Dual core1,66GHz CPU, 2Go RAM, Fast Ethernet 100Mbps
PC Dell	Video server	Dual core 3GHz CPU, 1.9Go RAM, Fast Ethernet
LinkSys	Access Point	802.11b/g wireless interface, Fast Ethernet

Figure 5: SESSAMO running on Nokia 770

3.5 Performance Evaluation

In order to evaluate the performance of our session mobility solution we consider the different time instances illustrated in Figure 6. We are interested in measuring the following periods of time: session handover delay (tshd), session overlap time (tso) and starting time of the player (tstp).

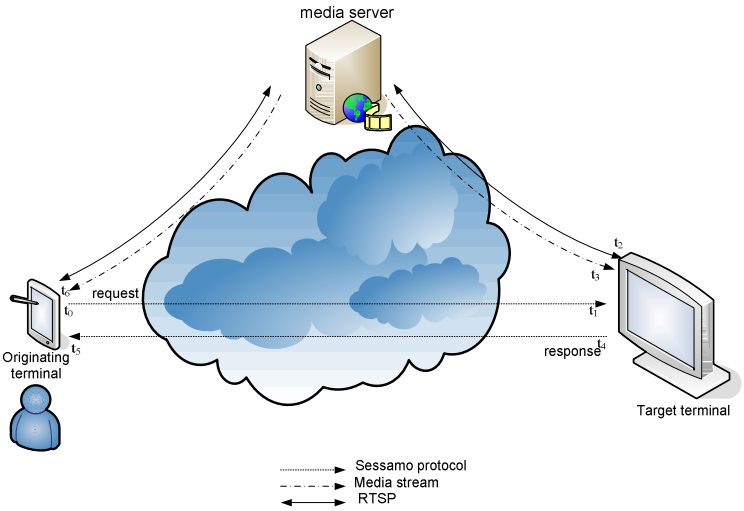

Figure 6: SESSAMO timing

Hereafter the list of the most important time instances used in measuring the performance metrics stated above, these instances are picked up using a network analyzer (wireshark [10]):
t0: the instant of time at which the user pushes the “transfer session” button.
t1: the time instant when SESSAMO request message arrives to TD.
t2: the time instant when TD starts the service negotiation with the video server.
t3: the time instant when RTP flow is received by TD.
t4: the time instant when SESSAMO response is sent to OD.
t5: the instant at which SESSAMO response is received by OD.
t6: the time instant when OD receives the last packet of audio/video.

t_{shd} is equal to $t_0 - t_3$ and it corresponds to the period of time that goes from the instant t_0 when the user pushes the button "session transfer" until the time instant t_3 when the video packets start arriving to TD. In practice, this period of time is difficult to measure because it involves the clocks of different terminals. Nevertheless, we propose the following approximation:

$$t_{shd} = t_3 - t_1 + RTT/2.$$

As far as the video session overlap time t_{os} is concerned, it indicates the duration of the video sequence that will be played in both devices prior to finish the original session. As the starting time of the session on the TD is equal to the instant of pushing the transfer button, t_{so} corresponds to $t_6 - t_0$. Finally, the starting time of the RTSP player t_{stp} is interesting because it gives as an indication about the effect of this operation on session transfer performance. It should be noted that player launching is not related to the mechanism of the mobility itself, but depends on the operating system and hardware capabilities of the terminal. Indeed, the starting time of each RTSP player varies considerably. It goes from some fractions of second to several seconds. t_{stp} can be obtained by means of $t_{stp} = t_2 - t_1$.

The experimentation was conducted ten times for each scenario and the mean value of each metric is calculated.

3.5.1 Make before break scenario

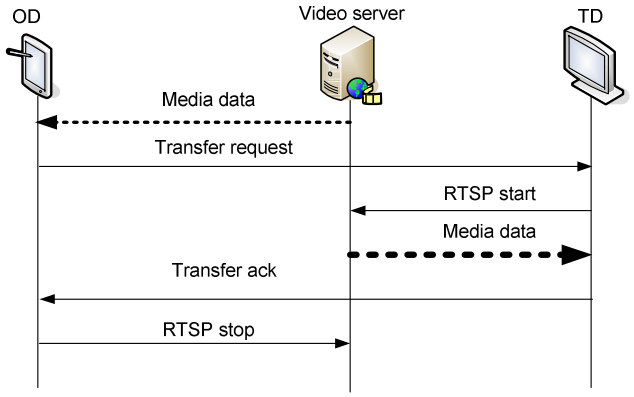

Figure 7: make before break scenario

Make before break approach as illustrated in Figure 7 consists of establishing the session on the TD before the acknowledgement is sent back to the OD. Acknowledgement here means that data traffic is arriving and the session was transferred with success. The advantage of this scheme is the guarantee that the stream is really received by the OD, if this latter can not establish the session with the video server, the acknowledgment will not be sent back and the session will not be terminated on the OD.

Table 2: make before break results

Delay	value in ms
Session handover delay	453
Video session overlap	455
VLC starting time	442

Table 2 summarizes results of session transfer. The handover delay is short and the transfer is almost instantaneous for the user. The disadvantage of this scheme is that video session overlap is high. To be noted that the starting time t_{stp} of the RTSP player represents 97% of the total session handover delay.

3.5.2 Break before make scenario

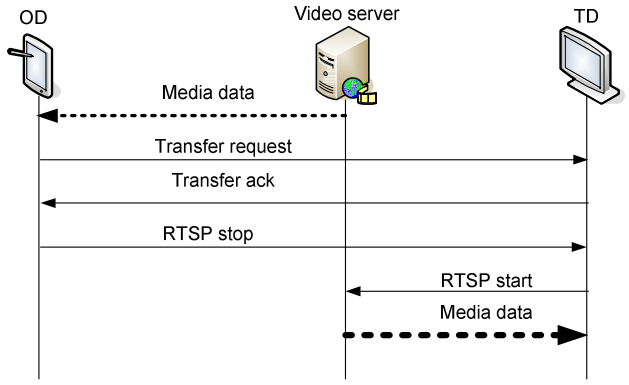

Figure 8: break before make scenario

In break before make scenario (see Figure 8) the acknowledgment is sent immediately after reception of transfer request, and subsequently the session is terminated on the OD. The advantage of this approach is that video overlap is reduced compared to the previous scheme as shown is Table 3. As for session handover delay is almost equal to the make before break handover delay since the principal cause of this delay is VLC starting time with more than 97%. The drawback of this scenario is that a silence time occurs during the transfer operation, especially when the handover delay high.

Table 3: break before make results

Delay	value in ms
Session handover delay	499
Video session overlap	9
VLC starting time	487

3.5.3 Synchronization issue

An ideal session mobility mechanism should provide perfect synchronization between the two devices. In other words TD should resume the session exactly from the same instance the user has triggered transfer button. This problem is not noticeable in small networks as in the case of our testbed where the delay between the different entities is negligible. But when it comes to large networks, congested networks or busy servers, the delay can be significant. Therefore, the overlap will be important in case of make before break and silence time will be high in case of break before make. In such conditions, make before break approach seems to be more adequate, because at least the user can continue watching the media on the old device until the service is established on the new one. As for video session overlap, it can be compensated by seeking the video in a farther point compared to the button push point. This difference in time should be equivalent to the estimated handover delay.

4. RENEGOTIATION OF QOS PARAMETERS

The challenge now is to cope with the variety of mobile devices. Indeed, we can find in the market different devices that have different screen sizes, CPU powers, operating systems, network interfaces, supported codecs, battery powers…etc. A server offering a given service has to be capable of satisfying each device according to its own capacities. Signaling protocols such as RTSP and SIP transport in their message payload the QoS constraints imposed by clients and their capacities. Media servers transcode or re-quantize data content accordingly. It is obvious that the media adaptation is a must in case of session mobility as we transfer the flow from one terminal to another. Nevertheless, we tackle the problem of QoS management in session mobility in different phases. The first phase is the negotiation that takes place at the beginning of the session; the second one is the adaptation during the session when some parameters change in the same terminal such as battery level and connection rate. The third phase is the re-negotiation when transferring the session. The particularity of our work resides in the manner these tasks are achieved by means of SDPng/MPEG-21 and RTSP. In particular, QoS service adaptation is not based on a classical approach where server adapts the flow without any participation from the client. Indeed, in our approach the client drives the server to obtain the most suitable QoS.
It should be noted that adaptation to network conditions is not the subject of our study. We focus on the parameters that are related to the terminal itself. If the quality of the video is degrading because of congestion in the route down to the terminal, other end to end techniques should intervene to adapt the media accordingly. Real time control protocol RTCP [2] is one of the possible solutions to control the flow using the periodic reports. These reports are sent back to give the server an idea about the available bandwidth along the route to the terminal.

4.1 Session Description protocol new generation (SDPng)

SDPng [11] is a description protocol for multimedia sessions. It is an application-independent framework transported by other signaling protocols such as SAP (session announcement protocol), RTSP, SIP…etc. the main innovation of SDPng compared to the conventional SDP [12] is its extensibility by using XML, which gives the possibility to describe the different terminal characteristics and user preferences. SDPng description is an XML document divided into 5 parts: capabilities, definitions, configuration, constraints and session information. We are interested in constraints parts, in order to accomplish multimedia adaptation. The constraints section allows expressing constraints on combination of terminal configuration. This feature is intended for specialized devices with strict limitations. To be noted that SDPng base specification is only a container for constraints and does not define them.

4.2 MPEG-21

Digital Items are defined as structured digital objects, including standard representation, identification and metadata. They constitute the fundamental unit of distribution and transaction within the MPEG-21 [13] framework.

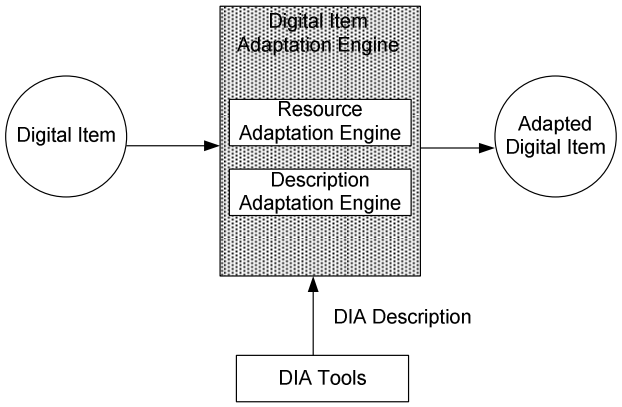

Figure 9: Concept of MPEG-21 DIA

In its seventh part, MPEG-21 defines a number of descriptors and tools to assist the adaptation of Digital Items (see the concept of DIA in Figure 9). Descriptors try to express on the one hand terminal capabilities, network descriptors, user characteristics and natural environment characteristics referred to as Usage Environment Descriptions (UED). On the other hand, it describes the high-level structure of a bitstream referred to as Bitstream Syntax Descriptions (BSD). It is important to underline here that only tools used to guide the adaptation engine are specified by the standard, as for the adaptation engines themselves are left open to various implementations. In order to be independent and open for novel developments, MPEG-21 does not specify any relationship with existing transport mechanisms.

4.3 Integration of MPEG-21 DIA in SDPng

Guenkova et al [14] propose a harmonization between MPEG21 DIA and SDPng by embedding MPEG21 DIA into SDPng. The proposed mechanism allows the definition of system configurations, performance constraints and adaptation information within the scope of SDPng using the format of MPEG-21 DIA. Furthermore, the converged format enables the integration of session management and negotiation protocols that use such enhanced SDPng descriptions within an MPEG-21 compliant environment. Since the two standards are XML-based, this combination can take place easily by integrating MPEG-21 DIA namespaces to SDPng document. This idea came from the fact that SDPng currently specifies only a container of terminal characteristics and MPEG-21 is not matched to any transport mechanism.

4.4 QoS management

We propose in this work a scheme using SDPng/MPEG-21 in the context of mobile multimedia applications that works under the client-server paradigm. Here, SDPng and MPEG-21 are employed to specify the QoS requirements exposed by the client application at different stages of the session's life, specifically, QoS negotiation and renegotiation stages. Negotiation is the first operation that takes place before streaming starts. The purpose of this operation is to inform the server about the characteristics of a given device in order to serve an adequate coded stream. As far as renegotiation is concerned, we consider two cases where it can take place. The first one occurs when a session moves from one terminal to another (i.e. session mobility) which has different capabilities than the first one. The second one arises when a "stable" session taking place over a given terminal suffers from an unexpected deficiency (battery level is low, system resources overload…etc.). In both cases, the application has to adapt its behavior according to the new circumstances in order to maintain its operation at an acceptable QoS level. In classical approaches, the server decides about the adaptation procedures to be used, relying on the information reported by the client. This information essentially describes the instantaneous network state in terms of the end-to-end delay and the packet loss ratio. One should notice that the client does not directly participate with the server to take decisions about the adaptation process because it only reports its perception about the network QoS. Indeed, the server adapts its behavior under a "best effort" scheme hoping that adaptation will be the best choice for the client. Our proposition contrasts with classical approaches because it gives a more active role to the client.

4.5 Specification of Qos aspects for session mobility

In distributed multimedia applications, QoS management is implemented to provide the final user with acceptable service. A very important aspect of QoS management is QoS specification, which should be conveyed to the server. This specification is composed of a set of parameters designated to describe accurately the QoS requirements of a distributed application. As we stated above, the tools that we use in our work are SDPng and MPEG-21. These standards take into account the new generation of applications that operate in a highly heterogeneous mobile context, but at this time they are not completely defined. Here are the main interesting parameters of which degradation can seriously damage the running session:

- Display properties: it includes colors properties and resolution of the frame. Resolution can be determined by the size of the window displaying the video not necessary the full screen resolution.
- Battery level: when the battery reaches critical levels, power consumption has to be reduced, i.e. the server can eliminate the video stream and keep only the audio.
- Memory space: in case of overload in the terminal, memory space becomes insufficient to run all applications.
- CPU utilization: in case the terminal is running other applications simultaneously with the media session, the CPU utilization is increased as it is shared among all applications and consequently causes a degradation of the perceived quality of the stream.

Network interface bit rate: changing access network in ubiquitous mobile environment is a potential operation. With cross layer protocols, the application can be informed about an imminent handover as well as the new access network characteristics.

4.6 Renegotiation

When a session moves from one terminal to another with different characteristics, a renegotiation process is required (see Figure 10).

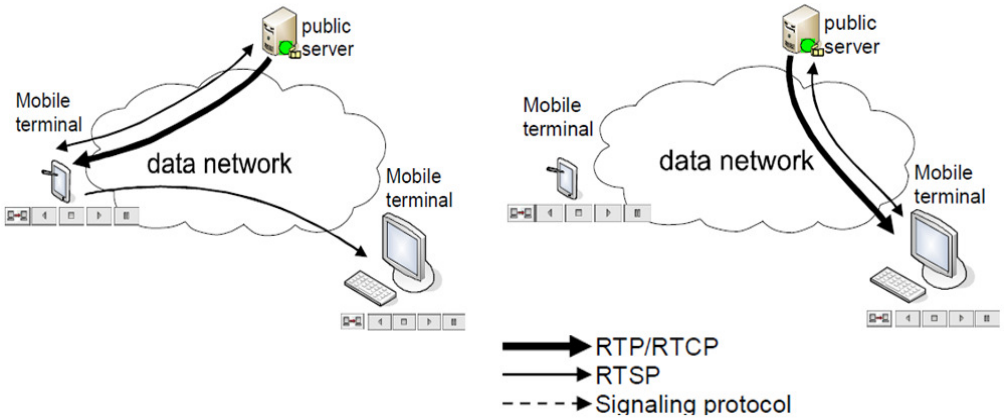

Figure 10: Session mobility with QoS renegotiation

Renegotiation of the new parameters is achieved by the new terminal itself. It specifies its constraints in the SDPng message and sends it to the server. In response, the server adapts the media flow according to the new constraints and transmits it to the new terminal.

4.7 Parameter change

The idea of QoS renegotiation within multimedia session is based on the exchange of RTSP messages containing information about the current status of the terminal capabilities. SET_PARAMETER is the RTSP message that we choose to conclude this task, because it is used to convey parameters related to the operation of the received service. SET_PARAMETER method is used to report QoS indicators within the current session in an SDPng like message from the client to the server as showed in Figure 11. Using this information, the server can decide whether an adaptation operation is required. It should be noted that during the session only parameters that has been changed are concerned by the notification.

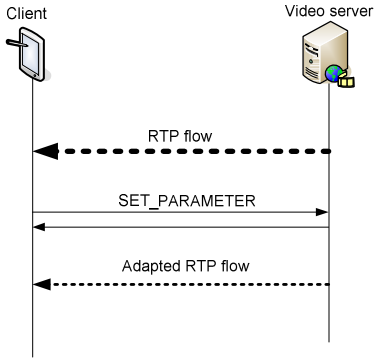

Figure 11: Message flow of renegotiation process

4.8 Adaptation to network conditions

The adaptation to network fluctuation does not need any renegotiation because the network is the cause of the problem not the terminal. As far as network communications are concerned, two approaches have been proposed to solve this problem. The first one proposes to enhance the network infrastructure by introducing resource reservation mechanisms. The second approach proposes to adapt the application to the available network resources. In the global Internet, adaptation of the flow from the application seems to be a more realistic solution. In order to implement the adaptation capability, the application should support some additional functionality. From a network perspective, a periodic feedback containing the current reception status of the flow allows the server to adjust the bit rate accordingly. A standardized protocol named RTCP (Real Time Control Protocol) is currently used to perform this task. The reported network state information allows the server to reduce or increase the bit rate in order to alleviate the losses in the sent flow. Of course, the losses rate will be reduced but the QoS perceived by the user will also be degraded, but in a controlled manner. There are several techniques allowing a server to adapt the flow to the available network resources, such as quantization, re-quantization [15], transcoding [16] [17], frame dropping, multilayer encoding [18]…etc.

5. CONCLUSION

Session mobility is an optional service that can be offered by the operator or even by content provider to their customers in order to give them more flexibility and portability regarding their interaction with the served media. In this paper, we proposed a new solution to support session mobility in video streaming services. SESSAMO is a lightweight protocol that operates between the concerned terminals and conveys the needed information for the target terminal to resume the session. SESSAMO has been implemented in a real life scenario using commercial mobile device. Moreover performance of the proposed solution has been conducted and the results show that this solution makes efficient and fast handovers. Indeed, achieving session transfer is a challenge in itself, but it has some “side effects” that should be treated as well. In fact when transporting the session from one terminal to another it is more likely that the latter has different hardware and software capabilities. Therefore, some measures have to be taken in order to adapt the served stream to the current terminal capacities. In this context, we proposed a mechanism to negotiate and renegotiate QoS parameters of the session by using SDPng and MPEG-21. Here, SDPng and MPEG-21 are employed to specify the constraints of the client at different stages of the session’s life. This description is integrated into the payload of specific RTSP messages following the required operation.

REFERENCES

[1] H. Schulzrinne, A. Rao, R.Lanphier, M. Westerlund, A. Narasimhan, “Real Time Streaming Protocol 2.0”, work in progress, IETF Draft, March 6, 2006

[2] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson,"RTP: A Transport Protocol for Real-time Applications", RFC 3550, July 2003.

[3] R. Shacham, H. Schulzrinne, S. Thakolsri, and W. Kellerer, “SIP session mobility”, Internet Draft (2006).

[4] M. Rawashdeh, A. Karmouch, "Seamless video handoff in session mobility over the IMS network", WoWMoM 2009.

[5] C. Yun, J Park, and Y. Lim “Session Mobility of IP Multimedia Subsystem (IMS) using Modified Assured (MA) Session Transfer”, 15th Asia-Pacific Conference on Communications (APCC) 2009

[6] Crocker, "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, UDEL, August 1982

[7] Live Networks inc., live555 Streaming Media,USA, 1999

[8] http://www.videolan.org/vlc/

[9] http://www.scratchbox.org/

[10] http://www.wireshark.org/

[11] D. Kutscher, J. Ott, C. Bormann "Session Description and Capability Negotiation ", IETF work-in-progress, draft-ietf-mmusic-sdpng-08, February 20, 2005.

[12] M. Handley, V. Jacobson, C. Perkins, “SDP: Session Description Protocol”, RFC 4566 July 2006

[13] A. Vetro, C. Timmerer, S. Devillers (eds.), ISO/IEC 21000-7:2004, "Information Technology - Multimedia Framework (MPEG-21) - Part 7: Digital Item Adaptation", October 2004.

[14] T. Guenkova-Luy et al, “Harmonisation of Session and Capability Description between SDPng and MPEG-21 Digital Item Adaptation”. IRTF work-in-progress, draft-guenkova-mmusic-mpeg-21-sdpng-00, Feb 2005.

[15] P. Assuncao and M. Ghanbari, “Post-processing of MPEG2 coded video For transmission at lower bitrates,” in Proc. IEEE Int. Conf. Acoustic, Speech, and Signal Processing, vol. 4, Atlanta,GA, May 1996, pp. 1998- 2001.

[16] A.Vetro, Charilaos Christopoulos, and Huifang Sun , “Video Transcoding Architectures and Techniques: an Overview”, IEEE Signal Processing Magazine, March 2003, pp. 18-29.

[17] A. Vetro, H. Sun, and Y. Wang, “Object based transcoding for adaptive video content delivery,” IEEE Trans. Circuits Syst. Video Technol., vol 11, pp. 387-401, Mar. 2001.

[18] R. Rejaie, Mark Handley, and Deborah Estrin, “Layered Quality Adaptation for Internet Video Streaming”, IEEE Journal on Selected Areas in Communications, vol. 18, No. 12, December 2000.