A solution for partial video voice over IP session transfer and retrieval

Sivasothy Shanmugalingam, Vincent Verdot, Noel Crespi, Paul Labrogere

A Solution for Partial Video Voice over IP Session Transfer and Retrieval

Sivasothy Shanmugalingam[1,2], Vincent Verdot[1], Noel Crespi[2], and Paul Labrogere[1]

[1] Alcatel Lucent Bell Labs France, Centre de Villarceaux, 91620, Nozay, France
[2]Institut Telecom, Telecom SudParis, 9 rue C.Fourier, 91011, Evry, France

sivasothy.shanmugalingam@alcatel-lucent.com, vincent.verdot@alcatel-lucent.com, noel.crespi@it-sudparis.eu, paul.labrogere@alcatel-lucent.com

Ubiquitous digital devices with rich media processing and networking capabilities open an avenue for enriching user experiences, especially in video voice over IP communication sessions because a communication session can be transferred partially to multiple devices that are in the vicinity of caller and (or) callee. Then, the transferred partial session can be retrieved back to the original place. For supporting partial video voice session transfer and retrieval, the solution needs a proper orchestration for the media channels. However, existing media orchestration mechanisms are separately and independently deployed on the user side (i.e. user agent) and network side (i.e. application server); thus, they increase the development and deployment complexity. Moreover, these solutions do not completely support seamless partial session transfer and retrieval at the caller and callee side. To overcome these problems, this paper proposes a solution for partial session transfer and retrieval (PSTR) that takes advantage of natural convergence of Web and communication services. Our solution supports any number of PSTRs at the caller and callee side without adding development and deployment complexity.

Keywords — session transfer and retrieval, Web-based communication system, media orchestrator.

I. INTRODUCTION

A communication session may consist of video or (&) voice streams in addition to signaling messages. The session can be transferred from one device to another (session mobility) on either the callee or caller side. This session can span (or transfer) over many devices for different media streams as presented in [1] (i.e. partial session mobility). Complete or partial session mobility can be initiated on the user side or in a dedicated mechanism in the network (for example a SIP Application Server (SIP AS)). Moreover, users prefer the option of retrieving their transferred media back during their mobility.

The solution for the above-mentioned complex scenario of Partial Session Transfer and Retrieval (PSTR) should include a proper orchestration mechanism for media streams, service/device discovery and negotiation [1]. An orchestration mechanism establishes right media streams across media devices regardless of originator (network side or users).

In this paper, we will focus on the orchestration mechanism. Existing solutions are divided into either user side [2] or network side [1] mechanism. Both user and network side approaches need to deploy third-party call control (3PCC) mechanisms. This means that for PSTR at the user side, two instances of 3PCC must be executed on the user side. For the PSTR by the network side, one instance of 3PCC should be executed in the SIP AS. Therefore, this separated deployment adds complexity in developing a solution.

Moreover, existing solutions do not perfectly support seamless session transfer and/or retrieval on both the caller and callee sides. For example, the solution in [1] only considers the PSTR at one side, not both sides (caller and callee).

In this paper, our previous work [8] is extended for partial video voice over IP (VVoIP) session transfer and retrieval. In [8], a central entity, the *Network box*, is hosted in the Web server to coordinate the media flow between users (callee and caller). In this paper, the network box manages also the media flow across different devices, regardless of the initiator (either user or network). The users establish or receive calls from/in their Web browsers. We depict these as communication widgets. In addition, possible target devices for session mobility are shown as widgets in their Web browsers and users can perform transfer and retrieval by drag and drop actions.

The organization of this paper is as follows. In Section II, we review the existing solutions for PSTR and illustrate their drawbacks. Then, our solution is discussed extensively in Section III from the architecture to the call flows. We conclude the paper with remarks on future work.

II. STATE OF THE ART

In this section, we review the existing solutions for PSTR based on Session Initiation Protocol (SIP). Then, we will show the weakness of the integration of universal plug-and-play (UPnP) devices with SIP user agent for PSTR.

A. Existing solutions for PSTR

The current solutions for PSTR do not provide a single orchestrator to support network-initiated and user-initiated PSTR. Therefore, we classify these solutions into two types: network-initiated, and user-initiated.

1) Network-initiated

In [1], the authors propose a SIP back-to-back user agent (B2BUA) application, called a sub-session controller (SSC), that performs partial session mobility based on a modified SIP mechanism (INVITE-based). This approach defines two new headers called *pst-to* and *pst-call-id*; these are used with the INVITE method to inform the user about the PST. If new headers are not implemented in the UEs, it is hard to benefit from this kind of service. In our approach, changes can be made easily because user agents (which we identify as caller and callee boxes) are downloaded at run-time [6] [7].

When a mobile node (MN) retrieves the session, MN sends the INVITE message to the correspondent node (CN). In this case, the SSC may not be aware of this retrieval. The MN closes the opened connections by sending the BYE message to any auxiliary devices that are involved in partial session mobility on the MN side. This logic should be implemented in the MN in addition to the SSC.

This approach [1] does not separate the different aspects (goals of PSTR and session initiation/modification). Instead, it uses (RE-)INVITE to inform users about session-transfer approval. Therefore, this approach adds complexity to UE development and further maintenance.

The SSC approach does not provide adequate partial session mobility at both MNs and CNs. In our approach, the network box performs all the media control for the PSTR. Our solution leverages the protocol [10] for media control and defines its own messages that instruct the transfer and retrieval between the network and the caller/callee boxes.

Finally, services developed based on the B2BUA method are encountered with the scalability problem [1] [8]. Our solution is effectively scalable, since each callee has a unique network box [7] [8].

2) User- initiated

RFC 5631 proposes two different transfer modes: session handoff (SH) and mobile node control (MNC); this RFC focus is on enabling the media transfer based on the end-point (SIPUA) [2].

For the SH mode, an MN sends a REFER request to a local device (SIPUA) that can participate in the session. For transferring partial sessions to many local devices, the MN should send multiple REFER messages. However, the sending of many REFER methods is not supported by the existing SIP standards. Therefore, [9] proposes a new entity called Multi Device System Manager (MDSM) that acts as a 3PCC controller between CN and local devices. As a result, MDSM encounters the same problems as the MNC.

In MNC, an MN implements the 3PCC for PSTR. When the MN and CN perform 3PCC, it will increase transfer/retrieval delay during the PSTR. This means that MN and CN need to send a RE-INVITE message for every session transfer and retrieval (i.e. new media end-point descriptions). If any of these events occur simultaneously on both sides, there will be a race condition as reported in [11]. Moreover, MN and CN should implement UA and B2BUA, and thereafter, MN and CN can support PSTR. Obviously, this approach is very complex for development and implementation. In [9], it is not explained how a user interface is designed to show the transfer and retrieval based on the 3PCC approach.

Similar to the SH mode, [3] and [4] present solutions for transferring and retrieving a partial SIP session over multiple devices using the modified REFER method. To split a SIP session, the authors in [3] propose a new header called "Mobility" and a new concept called "Association". In this approach (i.e. SSIP), MN establishes a session with CN. During the splitting, MN sends a REFER request to a free node with the mobility header. Later, the free node sends a new invite message with a mobility header to the CN. In this case, the CN should identify that there are many sub-sessions within a single session. For terminating the session, the CN sends an individual BYE message based on association.

The mobility header in the REFER method allows the referee to be informed about the session medium by the mobile node. The main drawback in the SSIP method is that it does not support the case when both, caller and callee side, start transferring the partial session. This means that session transfer and retrieval is considered only in the MN and not in the CN. In our approach, both end-points can perform session transfer and retrieval at any time; we explicitly define the free nodes based on media capabilities.

For another example, the SSIP concept is extended to support retrieval using the nested REFER method [4] [5]. To retrieve a transferred session back to the mobile node, four messages must be sent between the mobile node and the free node. In our approach, only two messages need to be sent between a mobile node (caller and callee boxes) and the network box.

B. Integration of UPnP devices and SIP UA

Integration of UPnP devices and SIP UA can be used for PSTR as shown in [12]. However, this solution supports only user-initiated PSTR where both SIP UAs (callee and caller) execute a 3PCC mechanism. In addition to drawbacks of MNC, this approach requires a large work investment for development of the proper linking of the SIP UA and the UPnP control point.

III. SOLUTION

Our solution consists of architecture, protocol and call flow for the PSTR.

A. Architecture

Our architecture has a *network box, caller box and callee box* [8] [10]. A *caller/callee box* is an end-point for callers and callees. Medium devices can send and/or receive the medium (audio/video) and are available and near to a callee and caller during a session. These medium devices can be divided into medium sources or sinks. In Figure 1, we did not show the medium device, but show the medium sink and source. A *network box* coordinates the media flow across the caller/callee box, and medium devices (and medium sinks and sources) according to user requests or its understanding of callee and caller context.

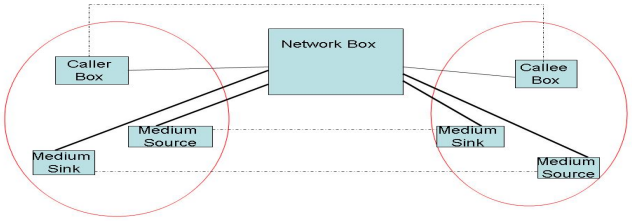

Figure 1: Architecture of the partial session transfer and retrieval. Thick lines show the connectivity established by the network box to medium sources and sinks.

In this sub-section, we describe each entity shown in Figure 1 and its functionality.

1) Network box

We proposed a *network box* in [8] for complete session mobility. In this paper, we leverage the *network box* for PSTR. While the *network box* manages the session across medium devices based on medium classification (see the next sub-section), it interacts with the caller and the callee via caller/callee box for PSTR.

2) Description of a Medium Device, Medium Sink and Medium Source

Generally, medium devices are capable of sending/receiving the media. Each medium device can play different roles such as managing the mediums (audio/video) or behaving as sink or source for a medium. These different roles are shown in the tree structure in Figure 2.

In addition, medium devices have a control interface, therefore, the network box can instruct medium devices for both sending and receiving.

We propose naming each media device based on the uniform resource identifier (URI) format and to access the control interface via Web Socket APIs. For example, a control interface of a medium device is ws://example.com/service or ws://ip:port/service/resource. Web sockets APIs facilitate simple integration between medium devices and the *network box* via Web.

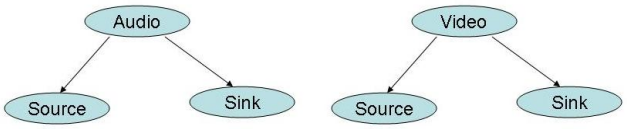

Figure 2: The classification of medium.

Detailed description of medium devices can help to differentiate the devices that can be source, or sink or both – fine-grained description. This will allow the medium device to be instructed based on the medium or the medium source/sink.

We propose a naming convention, based on URI, which is used to describe and to control services. The list of URIs for audio is shown below:

1. ws://ip:port/service/medium/audio
2. ws://ip:port/service/medium/audio/source
3. ws://ip:port/service/medium/audio/sink

Each device should implement three-message logic such as 'open', 'modify' and 'close' as proposed in [10]. The URI provides two details: the address of the device and which medium it supports.

Each media device has a user agent that can manage signaling and media. We consider the protocol stack shown in Figure 3, compared to a complete UPnP protocol stack that deals with addressing, discovering, description, controlling, eventing and presentation. Our protocol stack is lightweight in terms of processing compared to the UPnP stack that depends on Simple Object Access Protocol (SOAP).

Figure 3: Protocol stack for medium devices.

3) Caller Box/Callee Box

A caller box and callee box represents a caller and callee who establish a session using their Web browser. A caller box or a callee box is shown as a Web widget - in fact, each medium device can be represented as a Web widget as shown in Figure 4, which shows a sample user interface with the caller/callee widget, medium source widget, and medium sink widget. Therefore, users can perform a simple drag-and-drop action for PSTR.

We discuss later how the caller and callee boxes manage the protocol for PSTR in which the user is informed and in control during the network-initiated session transfer and retrieval later in sub-section III.C.1.

Figure 4: Sample user interface for partial session transfer and retrieval.

B. Protocol

Before diving into call flows, we describe our approach for developing the control protocol that will support PSTR. Our approach tries to separate the concerns. Therefore, we identify two protocols: media control [10] and our protocol called auxiliary protocol. The two protocols are dedicated to two different concerns; therefore, overall protocol development complexity will be reduced. The separation of concerns is one of the key principles of software engineering.

We define the auxiliary protocol by which abstractions facilitate carrying PSTR information between the network box and caller/callee boxes. This auxiliary protocol functions based on a request and response model that relies on TCP. All the abstractions in the auxiliary protocol are listed in Table 1.

TABLE 1: THE LIST OF ABSTRACTIONS FOR THE AUXILIARY PROTOCOL

Abstractions needed to initiate PSTR by a caller or callee	Abstractions needed to initiate PSTR by network box
Split (*URI*)	IsSplit (*URI*)
Splitted (*URI*)	YesSplit(*URI*)
NoSplit(*URI*)	NoSplit(*URI*)
Retrieve (*URI*).	IsRetrieve(*URI*)
Retrieved (*URI*)	YesRetrieve (*URI*)
NoRetrieval (*URI*)	NoRetrieve (*URI*)

URI can be either Medium or Medium Sink or Medium Source.

The abstractions above carry out the goals of transfer and retrieval. For example, if a caller/callee wants to transfer a partial session, they send a *Split* message to the *network box*. If the network accepts the transfer, it will send the *Splitted* message. Otherwise, the network box sends the *NoSplit* message. Each abstraction has a single argument in the form of URI (as mentioned in III.A.2). We show in the next sub-section how the media control protocol and the auxiliary protocol work together.

C. Call Flows

This section presents two different scenarios: Network-initiated and user-initiated PSTR. These call flows are composed of media control protocol and auxiliary protocol.

We made one change in the caller/callee box during the transfer/retrieval for a particular medium compared to [8]. During the session transfer, the caller box or callee box considers that muteIn, and muteOut are true [8]. If a session is retrieved, muteIn and(/or) muteOut become false.

Since transfer or retrieval can be initiated by users or the network, users are always privileged. All the activities initiated by the network side should be approved by the user. This means that the user keeps complete control.

1. Network-initiated partial session transfer/retrieval

Since the user always has the control, the *network box* requests user approval for PSTR. For this purpose, we use two different messages such as *IsSplit* and *IsRetrieve* for the PSTR. Figure 5 shows details of a session established between *caller* and *callee*. Once the media channel is established, *network box* asks *caller* for PST by sending the *IsSplit* message. If *caller* accepts, the session will be transferred as indicated in the *IsSplit* message. To transfer a session partially, *network box* sends the *open* message to the new device and modifies media parameters to the caller box by sending the *describe* message. Moreover, we show another interaction on the callee side. In this case, *network box* asks for permission of the callee box by sending the *IsSplit* message. When *callee* accepts the request, session is transferred partially at the callee side.

Similarly, *network box* requests caller to retrieve the session by sending the *IsRetrieve* message. When caller accepts the request, the session is retrieved. For retrieval, the *network box* closes the existing connection and updates media parameters with the *caller box*. If a user (caller or callee) wants to stop the session, he/she can send a close request to the network box that ensures disconnections with all the devices involved - including *caller and callee boxes*.

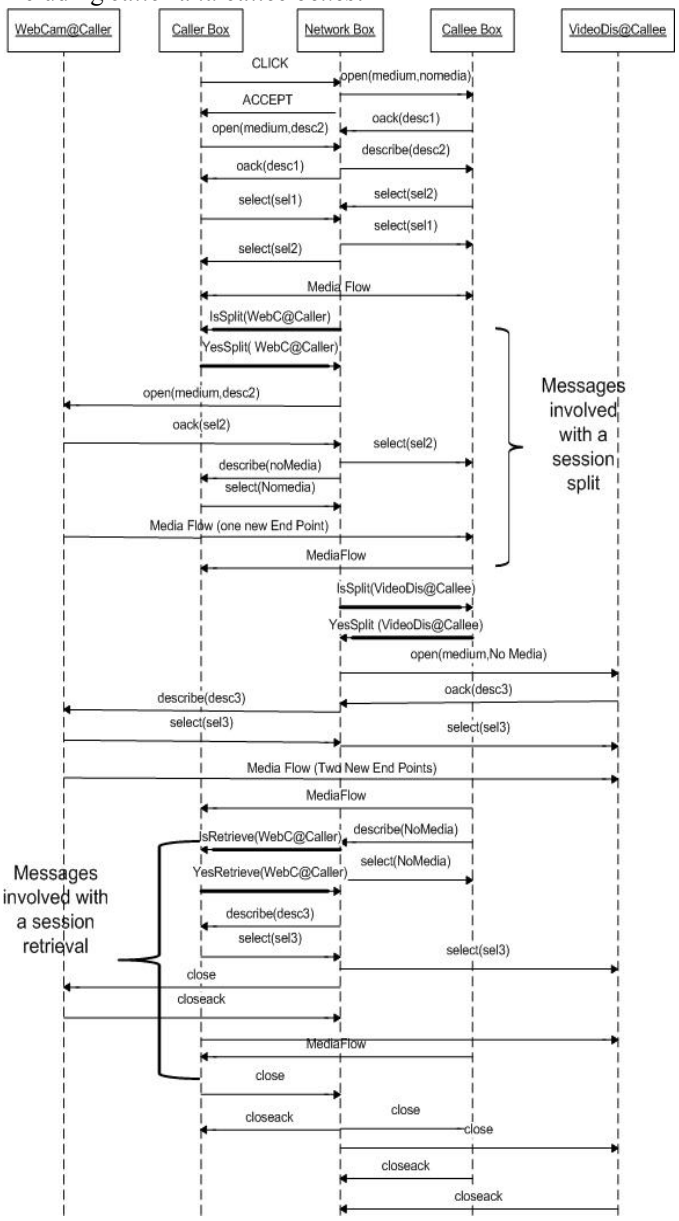

Figure 5: A call flow for network-initiated PSTR. Users are in control for transfer and retrieval. Here users will not initiate PSTR.

2. User-initiated partial session transfer/retrieval

For this scenario, a user issues two commands such as *Split(URI)* and *Retrieve(URI)*. Depending on the URIs, *callee/caller box* and *network box* perform media control. For example, if a URI refers to a medium source in a transfer request, the network box sends an *open* request to the URI mentioned (i.e. medium source) and a *description* message to

caller/callee box. Then, medium source accepts the request; and *network box* sends a *describe* message with the nomedia description to *callee/caller box*. Later, the *network box* sends the *Splitted* message to *callee/callee box*. This example call flow is shown in Figure 6 and includes WebCam@caller, caller box and network box entities.

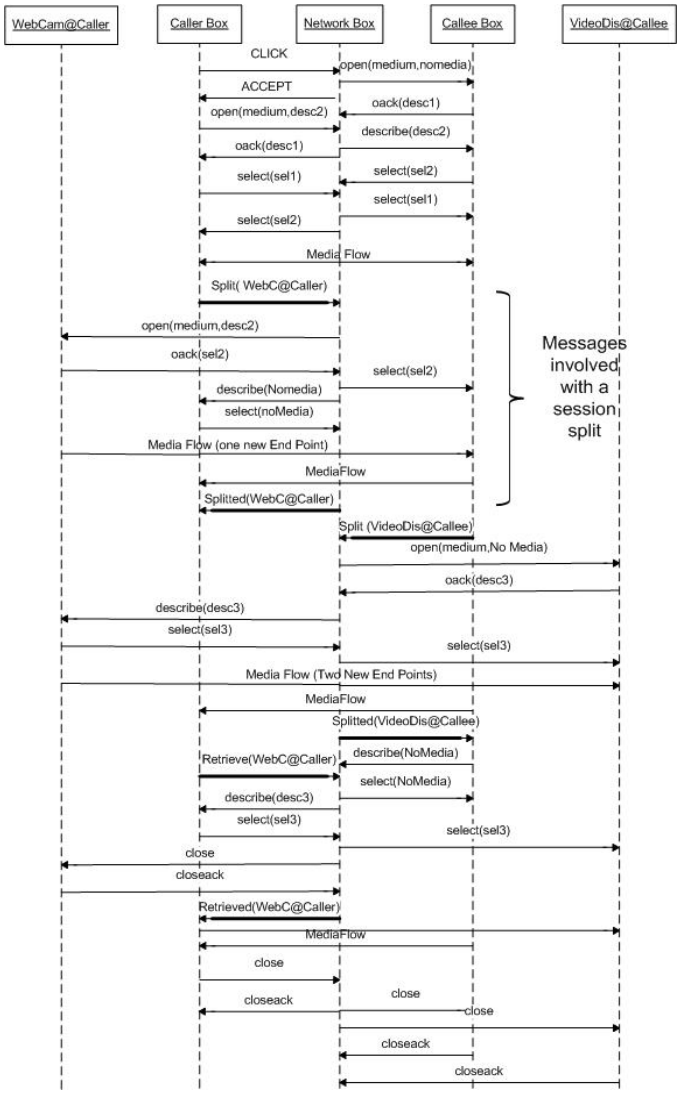

Figure 6: A call flow for user-initiated partial session transfer and retrieval.

If a URI refers to a medium sink, the *network box* opens a connection with the *medium sink*. If the *medium sink* accepts, the network box sends the *Splitted* message back. If the caller/callee box receives the *Splitted* message, the caller/callee box sends the *describe* message with nomedia. This interaction is included in Figure 6 that spans between VideoDis@callee, callee box and network box.

If a URI in a *Split* message refers to a medium, the network box sends an *open* message to the medium device. When the *medium device* accepts the request, the network box sends back the *Splitted* message. After sending the *Splitted* message by the network box or receiving the *Splitted* message by the caller/callee box, either end sends a *describe* message in order to make MuteOut and MuteIn true. Corresponding call flow is not included in Figure 6.

IV. CONCLUSION

Based on a single media orchestrator - *the network box*, we have presented our architecture for PSTR across multiple devices. This architecture facilitates network-initiated and user-initiated PSTR at the caller and callee side. It means that any number of transfer and retrieval can be performed by a *network box* and (or) users within a single session.

The complexity for developing this solution is reduced in two ways: 1) via a single media orchestrator at the network box and by 2) a signaling protocol design. We separate the signaling protocol into media control and auxiliary protocol based on the software engineering approach 'separation of concern'.

Since our solution is based on the Web, an end user (callee/caller) can transfer and retrieve the partial session by the drag-and-drop of widgets in their Web browser. This widget-based approach will increase the user's experience.

We propose a new user agent for media devices. The proposed changes need an agreement at the UPnP/DLNA level for rolling out this feature in all devices. In the future, by implementing the proposed system, we will be able to feed our requirements to the UPnP/DLNA forum.

REFERENCES

[1] Jasper Aartse Tuijn and Dennis Bijwaard. 2008. Spanning a multimedia session across multiple devices. Bell Lab. Tech. J. 12, 4 (February 2008), 179-193. DOI=10.1002/bltj.v12:4 http://dx.doi.org/10.1002/bltj.v12:4.

[2] R. Shacham, H. Schulzrinne, S. Thakolsri, and W. Kellerer, "Session Initiation Protocol (SIP) Session Mobility," IETF RFC 5631, Oct. 2009.

[3] M.-X. Chen et al., "SSIP: Split a SIP session over multiple devices," Computer Standards and Interfaces, vol. 29, no. 5, pp.531-545, July 2007.

[4] Chen, M.-X. and Wang, F.-J. (2010), Session integration service over multiple devices. International Journal of Communication Systems, 23: 673–690. doi: 10.1002/dac.1109

[5] Sparks, R.: The SIP Referred-By Mechanism, IETF RFC 3892

[6] Boussard, Mathieu; Jabaud, Philippe; Le Berre, Olivier; Poussiere, Fabrice; Labrogere, Paul, "Communication hyperlinks: Call me my way," *Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference on* , pp.1-5, 26-29 Oct. 2009

[7] Shanmugalingam, S.; Crespi, N.; Labrogere, P.; , "My Own Communication Service Provider," Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2010 International Congress on , pp.260-266, 18-20 Oct. 2010.

[8] Shanmugalingam, S.; Crespi, N.; Labrogere, P.; , "User mobility in a Web-based communication system," Internet Multimedia Services Architecture and Application(IMSAA), 2010 IEEE 4th International Conference on , pp.1-6, 15-17 Dec. 2010

[9] R. Shacham, H. Schulzrinne, S. Thakolsri, and W. Kellerer, "The Virtual Device: Expanding Wireless Communication Services through Service Discovery and Session Mobility," in IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob'2005), Montreal, Canada, Aug.22-24, 2005, pp. 73-81.

[10] Zave, P. and Cheung, E. 2006. Compositional control of IP media. In Proceedings of the 2006 ACM CoNEXT Conference (Lisboa, Portugal, December 04 - 07, 2006). CoNEXT '06. ACM, New York, NY, 1-12.

[11] M. Hasebe et al , "Example Call Flows of Race Conditions in the Session Initiation Protocol (SIP)", IETF RFC 5407.

[12] Vilei, A., Convertino, G., and Crudo, F. 2006. A new UPnP architecture for distributed video voice over IP. In Proceedings of the 5th international Conference on Mobile and Ubiquitous Multimedia (Stanford, California, December 04 - 06, 2006). MUM '06, vol. 193. ACM, New York, NY, 2