Enabling Session Mobility in Full Mesh Conferencing Model

Wajdi ELLEUCH, Alain C. HOULE, and Samuel GUÉNETTE

Abstract— **Enabling service continuity from one user's device to another one while maintaining communication is a real need to maximize end-user service experience and to benefit from the capacities of each available device in the user environment. This service ubiquity can be realized by session mobility between devices. Many session transfer use-cases can be supported to enable different service modes and options within a two-party communication context. Otherwise, enabling session mobility in a multiparty conferencing, like the full mesh communication model, can be more complex. This work describes a solution to support session mobility options in the full mesh conferencing by extending existing abstract message protocol and to map them using Session Initiation Protocol (SIP). Introducing session mobility in full-mesh conferencing causes different extra message traffic depending on the used mode (Mobile Node Control Mode or Session Handoff Mode). At the end, we propose a comparative study to determine the message traffic for each mode depending on the number of participants and the events included in conferencing scenarios.**

Index Terms—**Full mesh multiparty conference, Mobile communication, Session mobility, SIP protocol**

I. Introduction

The last few years have shown a noticeable improvement of handheld devices capacities, enabling them to support large multimedia services, applications and IP-based wireless and wired connectivity. Users of theses devices can then easily move within a wireless domain and enjoy spatial mobility while using multimedia services. However, these wireless handheld devices are still limited in term of autonomy, display capacity, ease of use and computational power. Stationary devices, like PC or wireline IP-Phones, continue to be more adapted for multimedia services, but not spatial mobility. By enabling seamless service transition between all available user-devices, it will be possible to benefit from each one and consider them as a unique device (virtual device) [1] [2].

Today, most deployed services are session-oriented. As an example, web services require a client browser to first establish session and session-ID used by the website to track and to identify the client browser when navigating within the website. Some session-oriented protocols, such as HTTP, can fit with such need [3]. For other internet services, like Internet Multimedia Subsystem (IMS) [4], the session concept is more significant and session establishment needs to be negotiated and managed. Dedicating exhaustive signaling protocol vocabulary is useful for exchanging and reporting session state and session parameters between user's services. Thereby, the Session Initiation Protocol (SIP) [5] has been chosen by the Third Generation Partnership Project (3GPP) as its standard for session establishment in the IMS [4].

Ensuring ubiquity and seamless service transfer between different communicating devices can be supported by ensuring some mobility in the application-session layer. Solutions were presented to enable session mobility between multimedia applications within two-party communication [1] [2]. These solutions use SIP mechanisms to handle the media entity (SIP Session) but also the signaling entity (SIP Dialog) of the communication.

In the case of multiparty communication, moving session from one participant to another device might significantly affect other participants. In situations where users join the conference or leave it, the transferred session host needs to be informed about such events and conference coherence has to be maintained. Enabling session mobility in multiparty conference appears less obvious than in the two-party communication context.

Several approaches and topologies have been proposed for distributed Internet multiparty communication [6] [7] [8]. Some of them were centralized approaches using a conference server that carries out signaling/media-mixing between participants. Two centralized approaches exist: The first approach aims to enable one participant of a two-party communication to invite other users and to ensure media mixing and conference signaling. The second approach intends to dedicate conference media mixing and/or session's

Manuscript received Mars 31, 2007. This work was made possible by a financial support from Lagassé Communications & Industries.

W.Elleuch B.Ing, M.Sc.A is a Ph.D candidate at the department of Electrical Engineering and Computer Engineering at Université de Sherbrooke, Sherbrooke (Québec) CANADA. (e-mail: Wajdi.Elleuch@USherbrooke.ca).
A.C.Houle, ing. Ph.D. is a professor in the department of Electrical Engineering and Computer Engineering at Université de Sherbrooke, Sherbrooke (Québec) CANADA. (e-mail: Alain.Houle@USherbrooke.ca).
S. Guénette, ing. M.Sc.A is General Manager of M5T Corporation, Sherbrooke (Québec) CANADA (e-mail: Sam@m5t.com).

IEEE COMPUTER SOCIETY

signaling management to a central third-party machine. The major drawback of the first approach is that as soon as the central element leaves the conference, there is no way for remaining participants to continue conferencing. On the other hand, the second approach presents some complexity when moving from a two-party call to a multiparty conference.

Some decentralized approaches use the multicast techniques to create network links between participants [8]. The deployment of such a solution requires using multicast routers and can cause a burden on network traffic. Another approach for decentralized conferencing is to create direct application-layer links between peers (peer to peer relationship) and to develop a full mesh topology. The full mesh model reduces existing conference models weakness by bringing more flexibility in conference creation and membership adding/removing management. This paper focuses on the full mesh model and aims to introduce new solutions to enable session mobility support modes and operations.

On the rest of the paper, we introduce a new session based IP communication layering model. In the section III, we present the session mobility concept, components and requirements. In the section IV, we explain the full mesh conferencing model. Section V explores the use of the session mobility in full mesh by presenting the message flow for each session transfer mode and by defining the extended message protocol. The transfer failure problem is also included in this section. Section VI includes the mapping of the conference message protocol to SIP. Section VII introduces the stack architecture components that support such session mobility in the full mesh model. In last sections, we analyze the extra message traffic generated by the session mobility options, we expose obtained results and we conclude this paper.

II. Application layer Mobility

Previous research in application layer mobility [9] [10] [11] defined four mobility layers: terminal mobility, session mobility, personal mobility and service mobility. While terminal mobility allows a device to continue using services while moving between IP subnets, the three other layer mobility focus on providing service to the user while moving between available devices. In other words, two different visions were developed: whether focusing service availability at the terminal-level or lead service provision at the user-level.

The user-level mobility paradigm was introduced by the Mobile People Architecture (MPA) concept [12] to enable users to be uniquely identified and to be contacted from anywhere and using a variety of communication media. Each user is identified by a unique ID (Personal Online IDs – POID) and uses a personal proxy that can track and manage the movement of the user with or without his device. Since the introduction of signaling protocols such as SIP for Multimedia over IP services, users can be identified by their SIP URI address while their devices continue to be identified by their IP address. To represent the user level mobility, the layered

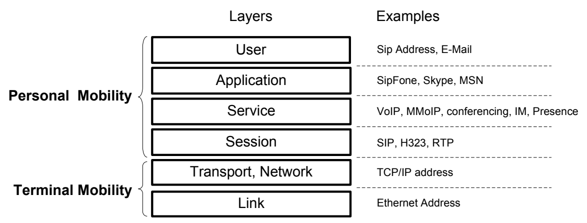


Fig. 1: Session based IP communication layering

communication model needs to put the final user on the top, as shown in Fig. 1.

The session concept is used to describe the media relationship used between participating peers. Session-based services are supported by applications located above the session layer [13]. The session layer plays the role of a middleware solution between the upper application and the lower network layers, as presented on Fig. 1. The aim of the presented layered model is to reduce the various session and dialog management overhead for the application/service designer. At the same time, the application layer can use the lower service layers either individually or by combining them into a single advanced service. As an example, we can mention the well-known presence service that, when combined with a simple instant messaging service, offers a powerful and unique service allowing users to exchange text messages with others while staying aware of their presence status. Also, it's important to note that service mobility is also bonded to the session mobility. Enabling session mobility between devices has an impact on all the upper layers.

III. Session mobility components and Requirements

A. Terminology and Requirements

Fig. 2 shows the architecture of our system. The Correspondent Nodes (CNs) are basic multimedia devices participating in a conference with the Mobile Node (MN). The MN is a device that has capabilities to handle session mobility. At anytime, the MN can move its active sessions to another available Local Nodes (LN). MN can use a service location directory to discover nearby available LN, as

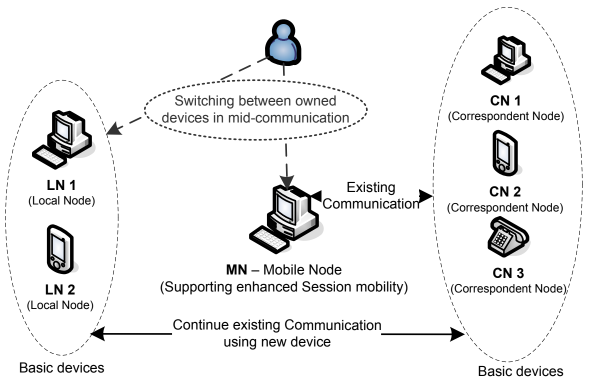


Fig. 2: General view of communication transfer

explained in [1] [2].

Participating nodes have to be full mesh enabled devices to take part in the conference. LNs can be basic devices or session mobility-enabled to preserve system interoperability and compatibility. All session mobility-enabled devices can act as MN by transferring session to other devices.

Session mobility also requires some conciliation flexibility for device capabilities differences and to enable LN to negotiate the most appropriate codec with remote participants. The sessions transfer between MN and each LN should fulfill seamlessness constraints, involving minimal disruption of transferred media flow. The transfer should not appear as a new call to the remote participants.

B. Use Cases

1) Transfer and retrieval

Transfer means to move the active session from the current device to one or more other devices. Retrieval means to transfer a session currently on a remote device back to the local device. For example, a user in videoconference communication with his handheld device enters new location where more adapted video display/acquisition devices are available. In this case, the user can transfer a video session to these devices. Before walking away, he can retrieve the video stream to his mobile device for continued communication.

2) Whole and split transfer

Session media may either be transferred completely to a single device or be split across multiple devices. For instance, a user may only wish to transfer the video portion of his session while maintaining the audio portion on his Personal Digital Assistant (PDA). Alternatively, he may find separate video and audio devices and may wish to transfer one media service to each of them. Furthermore, even the two directions of a full-duplex session may be split across devices. For example, a PDA's display may be too small for a good view of the other call participant, so the user may transfer video output to a projector and continue to use the PDA camera.

3) Transfer modes

Two different modes are possible for session transfer: Mobile Node Control mode and Session Handoff mode.

a) Mobile Node Control mode

In Mobile Node Control mode, a signaling session (dialog) is established with each device used in the transfer. The MN updates its session with the CN using Session Description Protocol (SDP) parameters to establish media sessions between the CN and each device, consequently replacing the current media session with the CN. The shortcoming of this approach is that it requires the MN to remain active to maintain sessions.

b) Session Handoff mode

A user may need to transfer a session completely because the battery on his mobile device is running out. Alternatively, the user of a static device that leaves the area and wishes to transfer the session to his mobile device, will not want the session control to remain on the static device when he is away. This could allow others to easily tamper with his call. In such case, Session Handoff mode, which completely transfers the session signaling and media to another device, is useful.

4) Types of transferred media

A communication session may consist of a number of media types, and a user should be able to transfer any of them to his chosen device. Audio and video are carried by standardized protocols like Real Time Protocol (RTP) [14] and negotiated in the body part of the signaling messages and encoded in some popular format like SDP [15]. Any example given for audio and video will work identically for text, as only the payloads differ.

IV. Full Mesh Conferencing model

A. General Structure

In the full mesh model, each endpoint pair is linked directly together with a separate dialog. Thereby, for N conference participants, each member needs to create and to manage (N-1) signaling dialogs (Fig. 3). Received (N-1) media flows have to be mixed locally by each node. The output media flows need to be duplicated and split through each conferencing session.

These dialogs were initiated, created and updated through message protocols, presented in the next section. The full mesh concept is more expensive in term of exchanged message cost than other conferencing models. However, it presents an ideal model where each participant is able to invite new members and drop out without affecting the remaining conference participants and without contacting a third-party machine like a conference server. Each node participating in full mesh conferencing has to manage both signaling and media traffic.

B. Protocol Messages

The basic protocol proposed in [6] was based on some abstract messages that can be sorted in three categories: request messages (JOIN, CONNECT, LEAVE, and UPDATE), response messages (OK, with the possibility to use Reject response for JOIN and CONNECT) and acknowledgment response (ACK) for messaging transaction

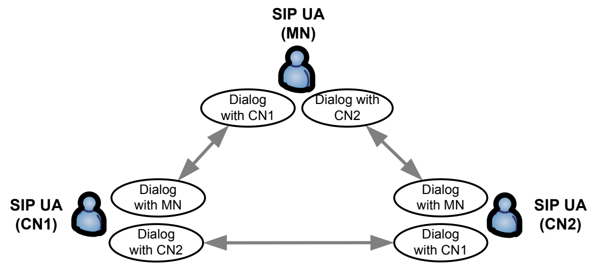


Fig. 3: Session interconnexion in full mesh

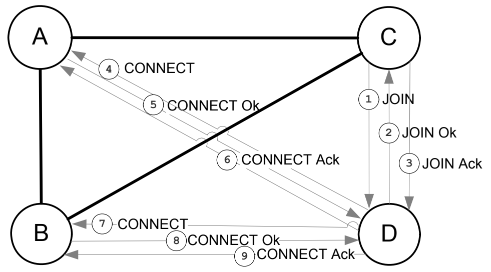


Fig. 4: Protocol messages in full mesh

initiated by JOIN or CONNECT requests.

While the JOIN message is used by conference members to add new users, the CONNECT message is used by invited users to establish communication with the remainder conference members. These two messages were based on the three phase messaging transaction (request/response/ acknowledgment). The UPDATE message can be used to inform each participant of new information about the conference membership list. The LEAVE message terminates the dialog. Fig. 4 shows an example of using the full mesh abstract protocol message to include the participant (D) to the established conference between A, B and C.

C. *SIP message mapping*

To implement the full mesh protocol in SIP, [6] provides a possible mapping of the full mesh protocol's abstract methods to concrete SIP methods. As both JOIN and CONNECT establish dialogs in the abstract protocol, they are both mapped to the SIP INVITE method. For similar reasons, LEAVE is mapped to either BYE or CANCEL, depending on the state of the dialog when it is invoked, and the UPDATE method can be mapped either to a re-INVITE or to a newly-defined SIP method (potentially, indeed, UPDATE [16]). The two subsequent phases of the connection process maps naturally: OK becomes a 2xx-class success response, REJECT becomes a 4xx, 5xx, or 6xx-class failure response, and ACK is ACK.

The implementation of the full mesh messaging protocol, as described in [6], requires introducing the same new headers.

TABLE I
MESSAGE PROTOCOL MAPPING TO SIP

Abstract message	SIP method	Added header
JOIN	INVITE	Conference-Id Conference-Member
CONNECT	INVITE	Conference-Id Invited-by
UPDATE	reINVITE UPDATE	Conference-Id Conference-Member
OK	2xx	Conference-Id Conference-Member*
ACK	ACK	Conference-Id Conference-Member*
LEAVE	BYE or CANCEL	Conference-Id
REJECT	4xx, 5xx or 6xx	Conference-Id

Each message needs to indicate the conference identification information by using a new *Conference-ID* header. Each full mesh conference will be then uniquely identified by an ID. This ID should be generated by the initial conference creator and by possibly using the same procedure as that used to generate the value of the *Call-ID* field.

The *Invited-By* header should be included on the CONNECT message, and used by the new added participant to specify the identity of the user that invited him (the basic SIP *Contact* header can be used to identify each participant). Finally, conference-Members list should be exchanged by participants and can be provided using new *Conference-Member* header field. Table I succinctly presents the message mapping and extra-header that should be added. The added headers, marked by (*), are optional.

V. SESSION MOBILITY IN FULL MESH

A. *Defining extended message protocol*

Basic messages used to create and manage full mesh conferencing need to be extended to support session mobility options. We propose four new message mechanisms that can be used to complete messages presented in section IV.B: MEDIA-JOIN/OK/ACK, JOIN-REFER/OK, CONNECT-REPLACE/OK/ACK and CONNECT-NOTIFY/OK.

The MEDIA-JOIN message is used in the Mobile Node Control mode when the MN needs to transfer a part of or the

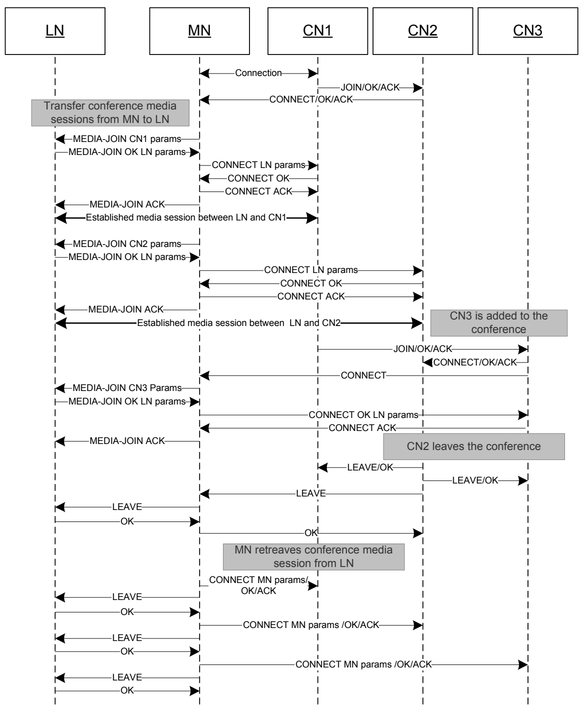


Fig. 5: Message flow for Mobile Node Control mode in full mesh

IEEE COMPUTER SOCIETY

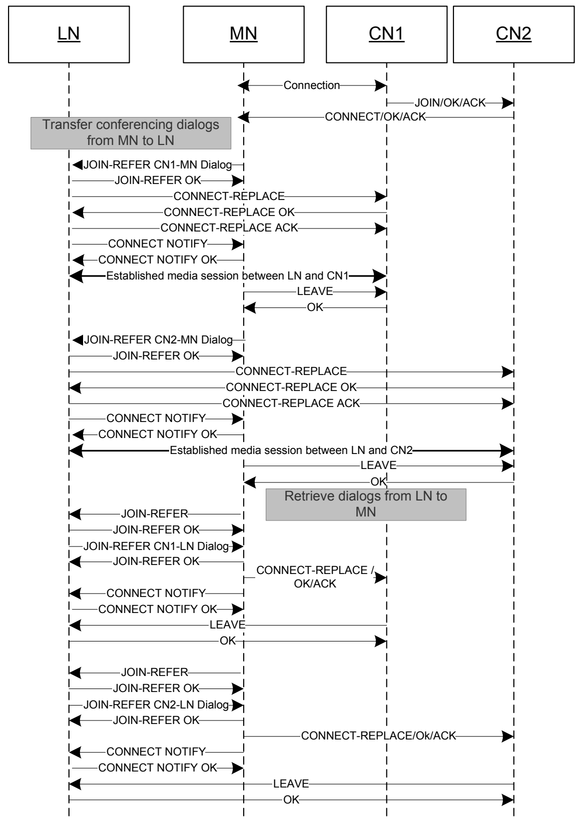


Fig. 6: Message flow for Session Handoff mode in full mesh

whole active media session to one or several LN. The MEDIA-JOIN mechanism is based on the three phases messaging transaction. It's used jointly with the CONNECT mechanism to permit media negotiation between LN and each CN within dialogs controlled by MN, as shown on Fig. 5. MEDIA-JOIN response can be whether positive (OK response), or negative (REJECT response).

The JOIN-REFER message request is used by the MN to refer the LN to replace its current session with each CN. The existing dialog parameter between MN and CN has to be included on the JOIN-REFER message and used by CN when requesting dialog substitution within CONNECT-REPLACE messages. The dialog replacement result can be reported to MN using CONNECT_NOTIFY message. If the notification indicates positive transfer response, MN can then proceed by terminating its dialog with the specified CN by sending a LEAVE message.

JOIN-REFER can also be used by MN as "nested REFER" to request LN for another REFER. This case of use can be useful to let MN retrieve dialog from LN on the Session Handoff mode.

This Dialog transfer procedure between MN and LN needs to be repetitively applied to include all CN participants.

B. Message flow analysis for session transfer modes

In this section, we explore the message flow to be deployed for each conference event and for both session mobility modes. We define four events that can generate message traffic for the conference: when a session is transferred, when a session is retrieved, when a user is added to the conference and, finally, when a user leaves the conference. Fig. 5 and Fig. 6 show respectively the message flow that should be exchanged between participant nodes for Mobile Node Control mode and Session Handoff mode.

The transferred full mesh conference sessions should be considered as a global transaction resulting from a set of individual session transfers between MN and each CN. We note that the Mobile Node Control mode is affected by adding/removing conference participant events. In this case, MN needs to forward each received CONNECT or LEAVE request to LN. MN will subsequently give response to these requests as soon as it receives a response from LN. Possibly, acknowledgements received from added participant, as a result of such transactions, need also to be routed to LN to maintain the MN-LN dialog coherence.

C. The session transfer failure problem

Conference Sessions transfer for both modes involves transferring all established CNi sessions with MN. During this session transfer, MN needs to proceed by transferring session by session for all (N-1) conference participant nodes. Sometimes, session transfer can fail for many reasons such as network problems or no common media codec found. In these

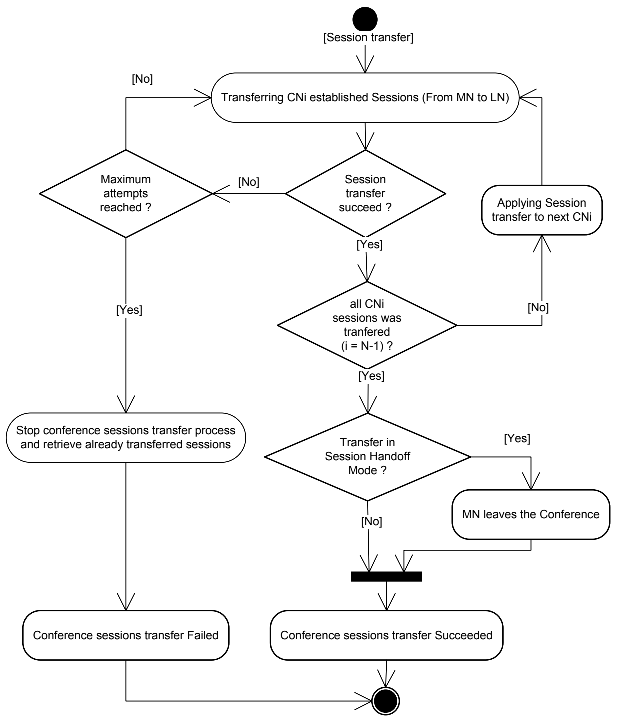

[Session transfer]
[No]
Transferring CNi established Sessions (From MN to LN)
Maximum attempts reached ?
[No]
Session transfer succeed ?
Applying Session transfer to next CNi
[Yes]
[Yes]
all CNi sessions was tranfered (i = N-1) ?
[No]
[Yes]
Stop conference sessions transfer process and retrieve already transferred sessions
Transfer in Session Handoff Mode ?
[Yes]
MN leaves the Conference
[No]
Conference sessions transfer Failed
Conference sessions transfer Succeeded

Fig. 7: Session transfer organization chart

TABLE II
EXTENDED MESSAGE PROTOCOL MAPPED TO SIP

Abstract message	SIP method	Added header
MEDIA-JOIN	INVITE	Conference-ID Conference-Member*
JOIN-REFER	REFER	Conference-ID Conference-Members*
CONNECT-REPLACE	INVITE	Replaces Conference-ID Invited-by* Conference-Members*
CONNECT-NOTIFY	NOTIFY	Conference-ID

cases, MN can proceed by retrying the transfer operation until reaching a maximum number of attempts. If session transfer fails, despite retrying, the session transfer process has to be canceled and already transferred sessions need to be retrieved. Fig. 7 shows the session transfer organization chart that can be used for managing the conference session transfer operation for both modes.

VI. MAPPING EXTENDED MESSAGE PROTOCOL TO SIP

In section V.A, we defined the extended message protocol that can be introduced to ensure session mobility in full mesh conferences. In this section we describe how this abstract protocol can be expressed in actual SIP messages.

The session transfer in mobile node control mode can be ensured by following the Third Party Call Control Flow I specified in [17]. This flow is recommended as long as CN immediately answers. The MEDIA-JOIN message can then be mapped to the INVITE SIP message. OK messages can be mapped to 2xx class success response and negative REJECT message will be 4xx, 5xx or 6xx class failure responses. The ACK acknowledgement remains the same.

JOIN-REFER request, used to initiate Session Handoff mode, can be mapped to SIP REFER message [18]. *Refer-To* and *Referred-By* [19] message headers included in the REFER message are used by LN to send the CONNECT-REPLACE request to CN. CONNECT-REPLACE can be mapped to INVITE SIP containing a *Replaces* header, as indicated in [20]. The *Replaces* header is used to identify the existing dialog that should be replaced by the new dialog.

Finally, CONNECT-NOTIFY can be mapped on the NOTIFY SIP message. Response abstract messages, such as OK or REJECT, are mapped identically as indicated in basic full mesh message protocol described in section IV.C.

To comply with existing full mesh message signalization protocol mechanisms, some headers need to be added to the mapped messages. The MEDIA-JOIN, mapped to SIP INVITE should include a *Conference-ID* header. In this way, LN will be able to understand that SIP dialogs, shared with MN, are linked to the same conference. In that case, LN needs to handle these sessions in conference mode where all output audio media streams, for example, need to be mixed. On the Session Handoff Mode, it's important that the CONNECT-REPLACE message sent from LN to CN includes a *Conference-ID* header that identifies the existing conference in which LN would like to participate. Some other headers like *Conference-Member* and *Invited-By* should be used to maintain coherence within the conference mechanism. The value of these headers should be transmitted by the MN on the JOIN-REFER message. Table II succinctly presents the message mapping and extra headers that should be added. The added headers, marked by (*), are optional.

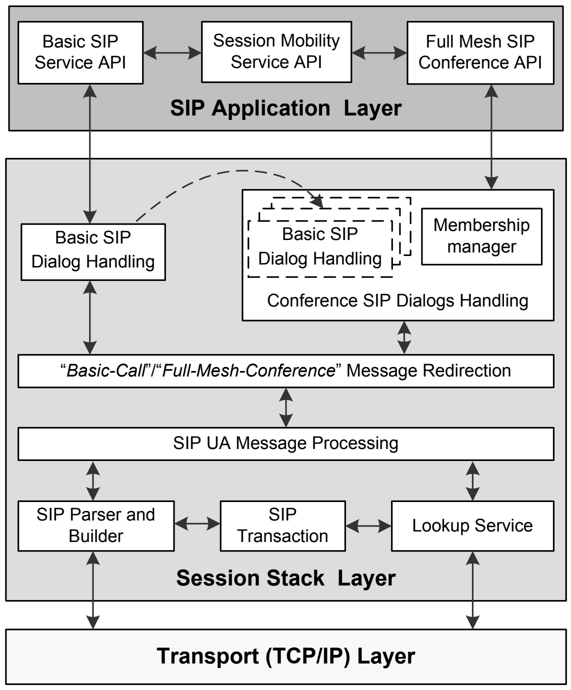
VII. STACK ARCHITECTURE COMPONENTS

Basic two-party call services can be represented by layering three component levels. These components are composed of the transport layer (TCP/IP), SIP Stack layer and SIP Application layer, as shown in Fig. 8.

Some mechanisms are generally included in commercially available SIP stacks to efficiently handle dialogs and to provide appropriate events to the SIP application layer. These mechanisms aim to reduce session management. In general, when creating a new communication dialog, a new handling manager is instantiated. Each SIP message identified as a part of the existing dialog will be automatically redirected to the appropriate dialog handling component.

In the case of a full mesh communication, each conference needs to maintain a set of dialogs, a piece handled individually be the *dialog handling manager*.

To support full mesh conference, it's possible to extend the SIP application layer by adding special processing for dialogs belonging to each conference. However, such a solution


Basic SIP Service API
Session Mobility Service API
Full Mesh SIP Conference API
SIP Application Layer
Basic SIP Dialog Handling
Basic SIP Dialog Handling
Membership manager
Conference SIP Dialogs Handling
"Basic-Call"/"Full-Mesh-Conference" Message Redirection
SIP UA Message Processing
SIP Parser and Builder
SIP Transaction
Lookup Service
Session Stack Layer
Transport (TCP/IP) Layer
Fig. 8: Components interaction in layered application

IEEE COMPUTER SOCIETY

reduces global service performance by introducing extra processes when each SIP message needs to reach the application layer before being identified as a member of the specified conference dialog. The proposed solution introduces a new component on the Stack layer to redirect messages directly to the appropriate Dialog Handler (basic or conference dialog handler). This component, called *Basic-call/Full-Mesh Conferencing Message Redirection*, should check the existence of a *Conference-ID* message header before redirecting message. For each new received *Conference-ID* identifier number, a new *Conference SIP Dialogs Handling* is created. The "Conference-Members" header could be analyzed by the *Membership manager* integrated within the *Conference SIP Dialogs Handling*. This handler can call a new instantiation of *Basic SIP dialog handling* component if a new member needs to be added to the conference.

Session mobility options for the two modes can be implemented on the Application layer when the underlying SIP Session Stack complies with the latest IETF SIP RFC. For example, *Replaces* header included in SIP INVITE or *Refer-To* and *Referred-By* headers included in SIP REFER should be supported by the *Basic SIP Dialog Handling* component.

We propose to add a session mobility service API as a separate component to permit its simultaneous use by both *Basic SIP Service API*, This ensures session mobility for the two-party communication model and for the *Full Mesh API Conference API* of the Multiparty full mesh communication model.

VIII. EXTRA SIGNALLING TRAFFIC

In this section, we evaluate the extra message traffic cost associated with each session transfer mode. Considering the unit cost *Cm* of a unique message, two different strategies can be adopted. The first strategy defines a set of events within the conference scenario. This scenario includes a number of transferred sessions (Nst), a number of added users (Nua), a number of removed users (Nur) and a number of retrieved sessions (Nsr). For initial N conference members, we calculate the cost of each scenario. In the second strategy, we use a single predefined scenario and we vary the number of participants in the conference.

For simplicity, protocol reliability mechanisms, which are based on repeated messages, are not considered here. Also, provisional responses (1xx-class for SIP case) are not included in this analysis. Special media renegotiations that occur when devices change codec will not be considered. This study is based on the message flow diagrams presented in Fig. 5 and Fig. 6.

A. Session transferring Cost

Mobile Node Control mode:

$$C_{st} = (N-1) * 6 * C_{m} \qquad (1)$$

Session Handoff mode:

$$C_{st} = (N-1) * 9 * C_{m} \qquad (1')$$

B. Session retrieving Cost

Mobile Node Control mode:

$$C_{sr} = (N-1) * 5 * C_{m} \qquad (2)$$

Session Handoff mode:

$$C_{sr} = (N-1) * 11 * C_{m} \qquad (2')$$

C. User adding Cost

Mobile Node Control mode:

$$C_{ua} = Nst * 3 * C_{m} \qquad (3)$$

Session Handoff mode:

$$C_{ua} = 0 \qquad (3')$$

D. User removing Cost

Mobile Node Control mode:

$$C_{ur} = Nst * 2 * C_{m} \qquad (4)$$

Session Handoff mode:

$$C_{ur} = 0 \qquad (4')$$

Scenario Cost for Mobile Node Control mode =

$$C_{senario} = Nst * Cst + Nua * Cau + Nur * Cur + Nsr * Csr \qquad (5)$$

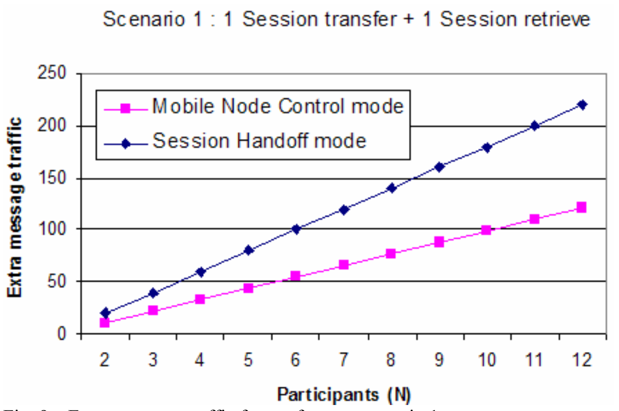


Fig. 9a: Extra message traffic for conference scenario 1

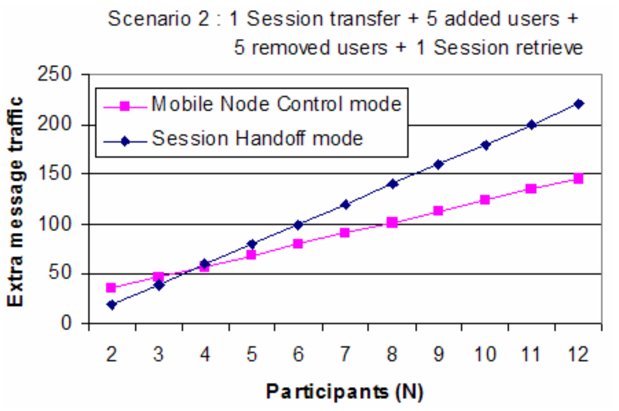


Fig. 9b: Extra message traffic for conference scenario 2

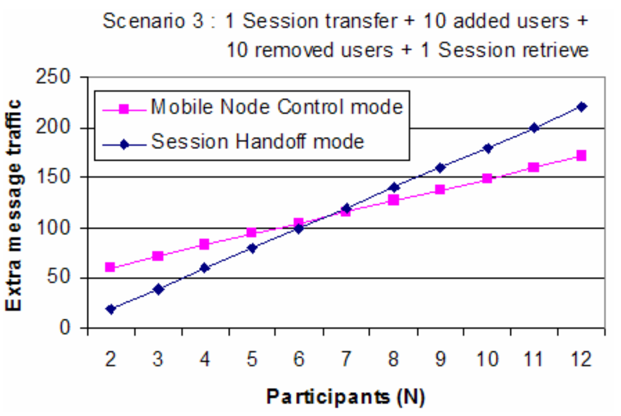


Fig. 9c: Extra message traffic for conference scenario 3

To simplify calculation, we consider that (Nst = Nsr) and (Nau = Nru). This means that scenario cost is equal to:

$$C_{senario} = Nst * (Cst + Csr) + Nua * (Cua + Cur) \quad (6)$$

We evaluate the scenario cost for Mobile Node Control mode (*Csena1*) and for Session handoff mode (*Csena2*). We replace the cost of each event by its corresponding value calculated in (1), (2), (3) and (4) depending on the scenario:

$$C_{sena1} = Nst*Cm*((N-1)*11 + Nua*5) \quad (7)$$

$$C_{sena2} = Nst*Cm*(N-1)*20 \quad (7')$$

By replacing the message cost (Cm) by the value "1", the scenario cost calculated on (7) and (7') will reflect the number of exchanged message of each scenario. On the other hand, (7) and (7') shows that the message cost ratio between the two transfer modes is independent on the Number of the transferred session (Nst). The Fig. 9a, 9b and 9c show the extra message traffic generated by each mode for three different scenarios where the number of added and removed users varies.

From Fig. 9a, it can be seen that the Mobile Node Control mode is more advantageous than the Session Handoff mode when no users are added or retrieved from the full mesh conference. As users are added or retrieved, the Session Handoff mode performs better than the Mobile Node Control mode for conferences with a small number of participants (Fig. 9b and Fig. 9c).

IX. Conclusion

We have presented a protocol and described mechanism and suited architecture that can complement existing full mesh solution to enable session mobility in this multiparty communication model. We also investigated the extra message flow generated by each session mobility mode.

Our future plans include optimizing session transfer operations and adding some improvement on session layer to be adapted for full mesh mode support.

References

[1] Shacham, R., Schulzrinne, H., Thakolsri, S., Kellerer, W., "The Virtual Device : Expanding Wireless Communication Services through Service Discovery and Session Mobility", Wireless And Mobile Computing, Networking And Communications, 2005.
[2] Shacham, R., Schulzrinne, H., Thakolsri, S., Kellerer, W., "Session Initiation Protocol (SIP) Session Mobility", IETF Draft, version 3, 2006.
[3] Song, H., Chu, H. H., Kurakake, S., "Browser Session Preservation and Migration", Poster Session of WWW Conference, Hawaii, U.S.A., 2002.
[4] 3GPP: TS 23.228: IP Multimedia Subsystem (IMS) (Stage 2)–Release 5, 2002, ftp://ftp.3gpp.org/Specs/archive/23_series/23.228/
[5] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Sparks, R.,Handley, M., Schooler, E., "SIP: Session Initiation Protocol", IETF RFC 3261, 2002.
[6] Lennox, J., Schulzrinne, H., "A Protocol for Reliable Decentralized Conferencing", International Workshop on Network and Operating System Support for Digital Audio and Video, 2003.
[7] Singh, K., Nair, G., Schulzrinne, H., "Centralized Conferencing using SIP", Proceedings of the2nd IP-Telephony Workshop (IPTel), 2001.
[8] Chu, Y. H., Rao, S. G., Seshan, S., Zhang. H., "Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture" In Proceedings of the ACM SIGCOMM Conference, pages 1–12, 2001.
[9] Henning Schulzrinne and Elin Wedlund, "Application-Layer Mobility Using SIP", Mobile Computing and Communications Review, Volume 4, Number 3, 2000, pp.47-57.
[10] Thai, B., Wan, R., Seneviratne, A., Rakotoarivelo, T., "Integrated Personal Mobility Architecture: A Complete Personal Mobility Solution", Mobile Networks and Applications, 2003.
[11] Di Stefano and C. Santoro, NetChaser: Agent support for personal mobility, IEEE Internet Computing (March–April 2000) 74–79.
[12] Appenzeller,G., Lai Petros, K., Roussopoulos, M., Swierk. E., Xinhua, Z., Baker, M., "The Mobile People Architecture", Technical Report, 1999.
[13] Yi Cui, Klara Nahrsetedt, Dongyan Xu, "Seamless User level Handoff in Ubiquitous Multimedia Service Delivery", Multimedia Tools and Applications Journal (ACM/Kluwer), Special Issue on Mobile Multimedia and Communications and m-Commerce, 2004.
[14] Schulzrinne, H., Frederick, R., Casner, S., Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", IETF RFC 1889, 1996
[15] Handley, M., Jacobson, V., "SDP: Session Description Protocol", IETF RFC 2327, 1998.
[16] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", IETF RFC 3311, 2002.
[17] Rosenberg, J., Peterson, J., Schulzrinne, H., G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", IETF RFC 3725, 2004.
[18] Sparks, R., "The Session Initiation' Protocol (SIP) Refer Method", IETF RFC 3515, 2003.
[19] Sparks, R., "The Session Initiation Protocol (SIP) Referred-By Mechanism", IETF RFC 3892, 2004.
[20] Mahy, R., Biggs, B., Dean, R."The Session Initiation Protocol (SIP) Replaces Header", IETF RFC 3891, 2004.

IEEE COMPUTER SOCIETY