Modeling of Multiparty Calls in SIP Networks

E. Pencheva, I. Atanasov

Key Words: *Open access to multiparty call control; SIP session state; labeled transition systems; behavioral equivalence.*

Abstract. *In IP-based multimedia networks, Open Service Access (OSA) is an approach to opening network by providing an interface that allows application developers outside the mobile domain to make use of network functionality and to receive information from the network. A Service Capability Server (SCS) is a functional entity that makes OSAstandard interfaces accessible by application and provides an abstraction of network protocol for application developers. As a gateway between applications and the network, the OSA SCS accomplishes mapping of OSA interfaces onto network protocols and vice versa. In the paper, we suggest an approach to modeling OSA SCS behavior in case of third party application control on multimedia sessions. The OSA SCS needs to maintain session state models that correspond both to the application view and Session Initiation Protocol (SIP) used for session management. The approach is described in a formal way using the theory of labeled transition systems. Defining bisimulation relation, we prove the behavioral equivalence between the suggested session state models and those of the standardized multiparty call state models as seen by OSA applications. To illustrate the behavior of OSA SCS, use cases for application control on SIP conferencing are provided.*

I. Introduction

The classical Intelligent Network (IN) is the first step to provisioning value-added services in telecommunication networks by centralizing service control in a dedicated node. The IN presents a closed business model. The service control node generally is a part of the network operator infrastructure and the operator owns the service logic programs and data. Convergence between IT applications and telecommunication services requires opening the network for third party application developers. This means that others than network operators and service providers are able to deploy services making use of network functionality.

There are several approaches to open networks such as JAIN, OMA service environment and OSA. JAIN, a name derived from Java Architecture for Integrated Networks, is an initiative that supports the development of applications by Java technology. The group of telecom operators and vendors called Open Mobile Alliance (OMA) defines service environment architecture, where the key component is the enabler. The enabler has public interfaces with particular functionality that can be used to develop and deploy services. Open Service Access (OSA) is a standardized open interface framework that allows third party applications to invoke communication functions in a public network. Network functions accessed by applications are called Service Capability Features (SCF). Application programming interfaces are defined for each OSA SCF exposed towards the applications. OSA is adopted as a service architecture in IP-based multimedia networks [1].

The OSA SCF interface towards the network is independent of the application logic i.e. the sequence of method invocation and responses is network independent. This interface independence is one of the problems in deploying OSA in convergent networks. There are no standards for the SCF interfaces towards the network, which can be based on Session Initiation Protocol (SIP), Customized Application for Mobile services Enhanced Logic Application Part (CAP), Intelligent Network Application Part (INAP), Megaco etc. The network element that provides OSA interfaces towards applications and control protocols towards the network is called Service Capability Server (SCS). It is considered as a special type of application server.

When the OSA SCF is deployed in IP-based multimedia networks, it has to expose SIP behavior towards the network. As to [1], OSA SCF has to support session state models that correspond both to the application view and SIP. *Figure1* shows the model for service control in IP-based multimedia networks, where the interface between the OSA SCF and the entity responsible for session management is SIP-based.

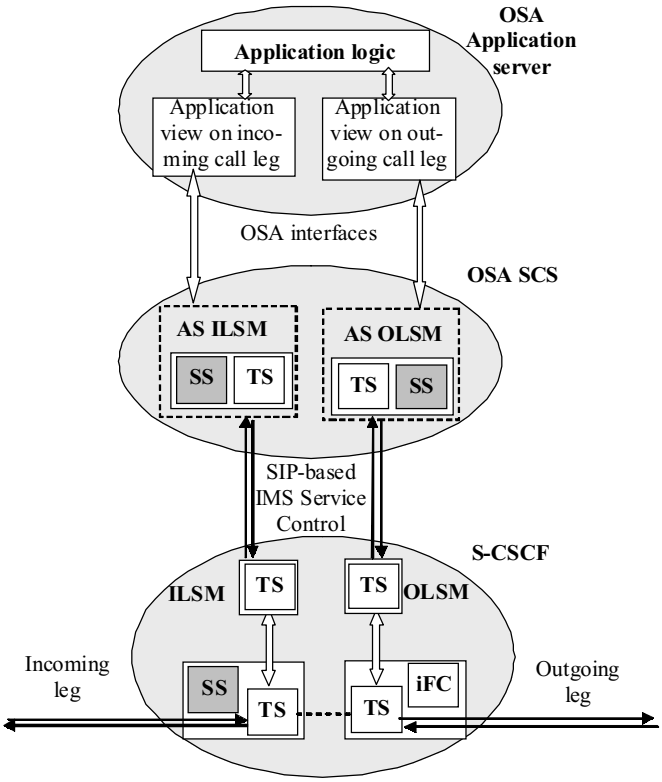

Figure 1. Service control in IP based multimedia networks showing transaction state (TS) and session state (SS)

Serving Call Session Control Function (S-CSCF) is a functional entity that is mainly responsible for session initiation, modification, and termination. For each SIP transaction a trans-

action state (TS) machine is triggered. The S-CSCF model describes the interactions with the originating party regarded as an incoming leg and terminating party regarded as an outgoing leg. These legs terminate the SIP transaction state machines in Incoming Leg State Model (ILSM) and Outgoing Leg State Model (OLSM) respectively. The S-CSCF maintains the session state (SS) also. Initial Filter Criteria (iFC) define conditions for triggering control logic in the application server. The S-CSCF communicates with the OSA SCS over SIP-based interface.

The components of OSA SCS include the AS-ILCM and the AS-OLCM. The AS-ILCM stores transaction state and session state. The AS-ILCM interfaces to the S-CSCF (ILCM) for an incoming leg. The AS-OLCM shall store transaction state and session state. The AS-OLCM interfaces to the S-CSCF (OLCM) for an outgoing leg.

The OSA application server hosts OSA applications. The main components include models that represent the application view on incoming and outgoing call legs and application logic. The Application Logic provides the service(s) and interacts between the AS-ILCM and AS-OLCM.

Examples of OSA application control in SIP multimedia sessions might be the following. In case of pre-arranged meet-me conference for a specified time period, parties can call in to the meet-me conference by dialing a special number. For each participant joining the conference, the application can decide to accept the participant into the conference. The application can also be notified when parties are leaving the conference. Another example is a prearranged add-on conference where the end user that initiates the call, communicates with the conference application via a web interface. By dragging and dropping names from the addressbook, the end-user adds parties to the conference. Also via the web-interface, the end-user can group parties in subconferences. Only parties in the same subconference can talk to each other.

While the transaction state models in SIP are standardized, there are no recommendations for SIP session state models. A mapping from OSA call control service to ISC interface is defined. Some of publications [2,3] concerning OSA SCS implementation focus on aspects related to the application programming interfaces, while other authors [4,5] discuss evaluation of conformance of the basic session control mechanisms of an IMS out of the application context.

Our research aims to suggest an approach to modeling session state in an OSA SCS. The session state model has to expose equivalent behavior to that of an OSA interface object as seen from application point of view. To prove the behavior equivalence we present the suggested SIP session state models in a formal way as *Labeled Transition Systems* (LTS) and use the concepts of bisimulation and homomorphism.

The rest of the paper is organized as follows. Section II briefly introduces the necessary notational background for labeled transition systems and behavioral equivalences. Section III presents the approach to modeling the SIP session state that conforms to the OSA application view of a multiparty call. Section IV describes models that represent the states of terminating and originating legs participating in a SIP session. Having in mind the implicit behavioral recommendations which are based on the available mapping of OSA Multiparty Call Control interface class methods onto SIP methods we provide adequate conditions on transformations in order to preserve the logic equivalence. In Section V, we provide use cases of OSA application control on SIP conferencing with centralized control. OSA Conference Call Control interface inherits Multiparty Call Control interface and the conference call state model resembles the multiparty call state model. The paper is concluded by sketching out an approach to evaluation of behavioral equivalence of OSA Generic call control and Data session control models, and state models in CAP.

II. Labeled Transition Systems and Behavioral Equivalence

Labeled Transition Systems are used when modeling interactive systems and studying their behavior. They provide formal tools for description of finite states models and for conformance testing [6,7].

Definition 1. [8] A *Label Transition System* (LTS) is a quadruple $(S, Act, \rightarrow, s_0)$, where S is a countable set of states, *Act* is a countable set of elementary actions, $\rightarrow \subseteq S \times Act \times S$ is a set of transitions, and $s_0 \in S$ is the set of initial states.

A labeled transition system $(S, Act, \rightarrow, s_0)$ is called *finitely branching*, if for any state $s \in S$, set $\{(a,s') , a \in Act \mid (s, a, s') \in \rightarrow\}$ is finite. The labeled transition systems we consider in the paper are only labeled transition systems that are finitely branching.

We will use the following notations:

– $s \xrightarrow{a} s'$ stands for the transition (s, a, s');

– $s \xrightarrow{a}$ means that $\exists\ s': s \xrightarrow{a} s'$;

– $s \overset{\mu}{\Longrightarrow} s_n$, where $\mu = a_1, a_2, ..., a_n : \exists\ s_1, s_2, ..., s_n$, such that $s \xrightarrow{a1} s_1 ... \xrightarrow{an} s_n$;

– $s \overset{\mu}{\Longrightarrow}$ means that $\exists s'$, such as $s \overset{\mu}{\Longrightarrow} s'$;

– $\overset{\hat{\mu}}{\Longrightarrow}$ means $\Longrightarrow$ if $\mu \equiv \tau$ or $\overset{\mu}{\Longrightarrow}$ otherwise.

Definition 2. Let $T = (S, Act, \rightarrow, s_0)$ and $T' = (S', Act, \rightarrow, s_0')$ are two labeled transition systems. Then

– $h: S \rightarrow S'$ is a *root preserving map* if $h(s_0) = s_0'$;

– A root preserving map is *surjective* if $h(S) = S'$.

Definition 3. Let $T = (S, Act, \rightarrow, s_0)$ and $T' = (S', Act, \rightarrow', s_0')$ are two labeled transition systems.

– $h: S \rightarrow S'$ is a *transition system homomorphism* if $h(s_0) = h(s_0')$ and $h(\rightarrow) \subseteq \rightarrow'$, where

$h(\rightarrow) = \{h(s) \xrightarrow{a} ' h(s') \mid s \xrightarrow{a} s'\}$;

– a homomorphism is *surjective* if $h(S) = S'$.

Definition 4. A *strong bisimulation* between two transition systems $T = (S, Act, \rightarrow, s_0)$ and $T' = (S', Act, \rightarrow, s_0')$ is a binary relation $U \subseteq S \times S'$, such that:

– s_0 U s_0';

– $\forall s \in S, \exists s' \in S'$: s U s' and
$\forall s' \in S', \exists s \in S$: s U s';

– if s U t, then, $\forall a \in Act$:

• $s \xrightarrow{a} s'$ implies $\exists$ t': $t \xrightarrow{a} t'$ and s' U t';

• $t \xrightarrow{a}$ 't' implies $\exists$ s': $s \xrightarrow{a} s'$ and s' U t'.

The labeled transition systems T and T' are bisimilar if there is a bisimulation between them.

III. SIP Session State Model for Multiparty Call Control

The open access to the multiparty call control for third party applications in OSA is provided by Multiparty Call Control SCF. A state model for MultiParty Call object is defined in [10]. This model reflects the application view on the session control and consists of three states as shown in *figure 2*. In the **Idle**$_{MPCC}$ state, there are no parties connected to the call, the **Active**$_{MPCC}$ state presents a call with one or more call legs connected to it,

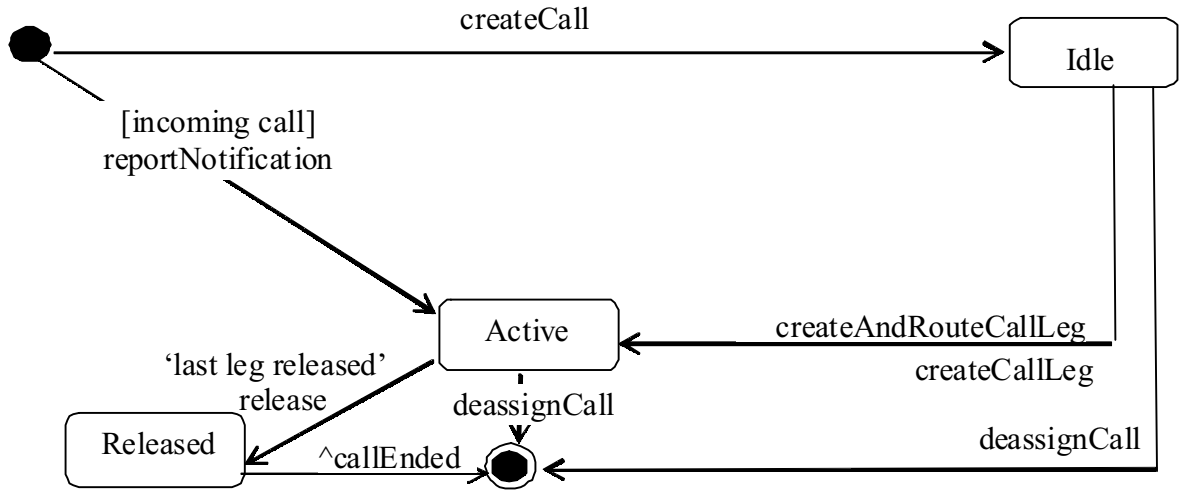

Figure 2. Application view on MultiParty Call object [3GRR TS 29.198-4-3]

Definition 5. Let $T = (S, Act, \rightarrow, s_0)$ and $T' = (S', Act, \rightarrow, s_0')$ are two labeled transition systems. The relation h: S $\rightarrow$ S' is called an *abstraction homomorphism* if h is a surjective transition system homomorphism satisfying the following:

$\forall s_1 \in S, a \in Act$ and $s_2' \in S'$, h $(s_1) \xrightarrow{a}$ ' s_2' implies

$\exists s_2 \in S : s_1 \xrightarrow{a} s_2$ and $h(s_2)=s_2'$.

The following theorem is proved in [9]:

Theorem 1. Two labeled transition systems are bisimilar iff they have a common image under abstraction homomorphism.

The strong bisimulation possesses strong conditions for equivalence which are not always required. For example, there may be internal activities that are not observable. The strong bisimulation ignores the internal transitions.

Definition 6. Two labeled transition systems $T = (S, A, \rightarrow, s_0)$ and $T' = (S', A, \rightarrow', s_0')$ are *weekly bisimilar* if there is a binary relation U $\subseteq S \times S'$ such as if s_1 U t_1 : $s_1 \subseteq S$ and $t_1 \subseteq S'$ then $\forall a \in Act$:

– $s_1 \overset{a}{\Rightarrow} s_2$ implies $\exists t_2 : t_1 \overset{\hat{a}}{\Rightarrow}' t_2$ and s_2 U t_2;

– $t_1 \overset{a}{\Rightarrow}' t_2$ implies $\exists s_2$: $s_1 \overset{\hat{a}}{\Rightarrow} s_2$ and s_2 U t_2.

and in the **Released**$_{MPCC}$ state, all parties are disconnect and the application can only gather information for the call. From the network point of view, in the **Released**$_{MPCC}$, the **SIP** session control is impossible, so in the formal description of the models using labeled transition systems, both **Idle**$_{MPCC}$ and **Released**$_{MPCC}$ states can be labeled by the same label.

Let us define $T_{MPCC} = (S_{MPCC}, Act_{MPCC}, \rightarrow, s_0)$ as labeled transition system representing the OSA application view on the MultiParty Call, defined for Multiparty Call Control SCF, where:

– S_{MPCC} = {Idle$_{MPCC}$, Active$_{MPCC}$ }

– Act_{MPCC} = {createAndRouteCallLeg, reportNotification, release, callEnded}

– $\rightarrow$ = {Idle$_{MPCC}$ createAndRouteCallLeg Active$_{MPCC}$,
Active$_{MPCC}$ release Idle$_{MPCC}$,
Idle$_{MPCC}$ reportNotification Active$_{MPCC}$,
Active$_{MPCC}$ callEnded Idle$_{MPCC}$}

– s_0 = {Idle$_{MPCC}$.}.

The transitions are labeled with the methods of the IpMultiPartyCall interface class. The reportNotification method is used to notify the application about events related to the call. The reported events are indicated as a notificationInfo parameter in reportNotification method.

We present a SIP session state model, maintained in an OSA SCS supporting Multiparty Call Control service, which is based on the behavior of a SIP statefull proxy during session initiation modification and termination. The model is formally described as a labeled transition system T_{SIP}, which is in a bisimilar relation with T_{MPCC}.

Initially we will present a simplified SIP session model with two states: **Idle**$_{SIP}$ (inactive SIP session), **Active**$_{SIP}$ (active SIP session).

Let $T_{SIP} = (S_{SIP}, Act_{SIP}, \rightarrow_{SIP}, s_0')$ be a labeled transition system representing the SIP session state model where:

– S_{SIP} = {Idle$_{SIP}$, Active$_{SIP}$ }

– Act_{SIP} = {INVITE, BYE}

– $\rightarrow_{SIP}$ = {Idle$_{SIP}$ INVITE Active$_{SIP}$, Active$_{SIP}$ BYE Idle$_{SIP}$}

– s_0' = {Idle$_{SIP}$}.

We define the following homomorphism h : $S_{OSA} \rightarrow S_{SIP}$ as follows:

h (Idle$_{MPCC}$) = Idle$_{SIP}$

h (Active$_{MPCC}$) = Active$_{SIP}$

h (Idle$_{MPCC}$ createAndRouteCallLeg Active$_{MPCC}$) = Idle$_{SIP}$ INVITE Active$_{SIP}$

h (Active$_{MPCC}$ release Idle$_{MPCC}$) = Active$_{SIP}$ BYE Idle$_{SIP}$

h (Active$_{MPCC}$ callEnded Idle$_{MPCC}$) = Active$_{SIP}$ BYE Idle$_{SIP}$

h (Idle$_{MPCC}$ reportNotification Active$_{MPCC}$) = Idle$_{SIP}$ INVITE Active$_{SIP}$.

Proposition 1. The labeled transition systems T_{MPCC} and T_{SIP} are bisimilar.

Proof. A mapping for Multiparty Call Control interfaces methods onto SIP methods is given in [11]. Following the mapping it is trivial to show that $Act_{MPCC} = Act_{SIP}$. Given the formal definitions of T_{MPCC} and T_{SIP}, and having that $Act_{MPCC} = Act_{SIP}$, implies by definition that *h* is an abstraction homomorphism of S_{MPCC} and S_{SIP}, which means that T_{MPCC} and T_{SIP} are bisimilar.

In multimedia networks where the session control is provided by SIP means, the procedure for quality of service negotiation and resource reservation takes place during session establishment. The SIP requests used for this purpose are PRACK and UPDATE. To refine the behavior of the SIP session, we bring in internal transitions in the composite Active$_{SIP}$ state and using the concept of week bisimulation we prove that the extended SIP session state model and the model representing the Multiparty Call states expose equivalent behavior.

To define refinement concept for a labeled transition system we introduce notions of state pre-condition and post-condition.

Definition 7. Given a labeled transition system $T = (S, Act, \rightarrow, s_0)$, the *pre-condition* of $s \in S$ denoted by $Pre(s)$ is $\{q \mid q \neq s, q \in S, q \rightarrow s\}$, and the *post-condition* of s, denoted by $Pos(s)$ is $\{q \mid q \neq s, q \in S, s \rightarrow q\}$.

Definition 8. Let $T = (S, Act, \rightarrow, s_0)$ be a labeled transition system and $s \in S$. A *refinement* of s denoted by $R(s)$ is a labeled transition system $R(s) = (S', Act', \rightarrow', s_0')$ where

– $s' \cap s = \varnothing$;

– $\rightarrow' \cap \rightarrow = \{ Pre(s) \cup Pos(s) \}$;

– $s_0' = \{s' \mid s' \in S, Pre(s) \subseteq Pos(s)\}$;

– $Act' = \{a' \mid p \xrightarrow{a'} q: p \in S', q \in S'\}$.

We refine the composite **Active**$_{SIP} \in S_{SIP}$ with **R(Active**$_{SIP}$**)** by adding states and transitions related to SIP requests and responses.

The refined SIP session state model is formally represented as a labeled transition system $T_{SIP}' = (S_{SIP}', Act_{SIP}', \rightarrow_{SIP}', s_0')$, where:

– S_{SIP}' = { Idle$_{SIP}$, SessionProgress, WaitPRack, PRacked, WaitUpdate, Updated, Wait180, Alerting, WaitAck, Established, Wait200$_{BYE}$ }

– Act_{SIP}' = {INVITE, BYE, 183, PRACK, 200$_{PRACK}$, UPDATE, 180, 200$_{INVITE}$, 200$_{UPDATE}$, 200$_{BYE}$, ACK}

– $\rightarrow_{SIP}'$ = {Idle$_{SIP}$ INVITE SessionProgress,
SessionProgress 183 WaitPRack,
WaitPRack PRACK PRacked,
PRacked 200$_{PRACK}$ WaitUpdate,
WaitUpdate UPDATE Updated,
Updated 200$_{UPDATE}$ Wait180,
Wait180 180 Alerting,
Alerting 200$_{INVITE}$ WaitAck,
WaitAck ACK Established,
Established BYE Wait200$_{BYE}$,
Wait200$_{BYE}$ 200$_{BYE}$ Idle$_{SIP}$ }

– s_0' = {Idle$_{SIP}$}.

Table1. Bisimilar relations between OSA multiparty call states and SIP session states

$s_1 \overset{a}{\Rightarrow} s_2 \mid s_1, s_2 \in S_{MPCC}$	$t_1 \overset{\hat{a}}{\Rightarrow}' t_2 \mid t_1, t_2 \in S_{SIP}$
Idle$_{MPCC}$ createAndRouteCallLeg Active$_{MPCC}$	Idle$_{SIP}$ INVITE SessionProgress, SessionProgress 183 WaitPRack, WaitPRack PRACK PRacked, PRacked 200$_{PRACK}$ WaitUpdate, WaitUpdate UPDATE Updated, Updated 200$_{UPDATE}$ Wait180, Wait180 180 Alerting, Alerting 200$_{INVITE}$ WaitAck, WaitAck ACK Established
Active$_{MPCC}$ release Idle$_{MPCC}$	Established BYE Wait200$_{BYE}$ Wait200$_{BYE}$ 200$_{BYE}$ Idle$_{SIP}$
Idle$_{MPCC}$ reportNotification Active$_{MPCC}$	Idle$_{SIP}$ INVITE SessionProgress, SessionProgress 183 WaitPRack, WaitPRack PRACK PRacked, PRacked 200$_{PRACK}$ WaitUpdate, WaitUpdate UPDATE Updated, Updated 200$_{UPDATE}$ Wait180, Wait180 180 Alerting, Alerting 200$_{INVITE}$ WaitAck, WaitAck ACK Established
Active$_{MPCC}$ callEnded Idle$_{MPCC}$	Established BYE Wait200$_{BYE}$ Wait200$_{BYE}$ 200$_{BYE}$ Idle$_{SIP}$

Proposition 2. The labeled transition systems T_{MPCC} and T_{SIP}' are weakly bisimilar.

Proof. To prove that two labeled transition systems are weakly bisimilar, we have to prove that there is a weak bisimulation relation between their states. Let U is a relation between the states of T_{MPCC} and T_{SIP}' where U ={(Idle$_{MPCC}$, Idle$_{SIP}$), (Active$_{MPCC}$, Established) }. The proof that the T_{MPCC} and T_{SIP}' are weakly bisimilar follows from the bisimilar relations $\overset{a}{\Rightarrow}$ and $\overset{\hat{a}}{\Rightarrow}'$ between the S_{MPCC}, and S_{SIP} shown in *table1*.

The refined SIP session state model can be extended with transitions that correspond to the SIP 3xx, 4xx, 5xx and 6xx responses following the mapping in [11].

4. SIP Session Incoming and Outgoing Leg Models

The Multiparty Call Control SCF application programming interfaces distinguish between the call (session) and the connection (party in call). Besides the model representing the application view on Multiparty call object, models of originating (ILSM) and terminating (OLSM) call legs are also defined [8]. The SIP incoming and outgoing state models (AS ILSM and AS OLSM) maintained in an OSA SCS, have to be synchronized with those representing the application view on originating and terminating legs.

A. Terminating party state model

From the application point of view, the terminating leg object may be in one of three states as shown in *figure 3*. In the Idle_{TERM} state, the Terminating leg object is created but not connected. In the Active_{TERM} state, routing information is gathered, the authority of the called party is verified, and after network indication of answer, a connection is established to the called party. In Releasing_{TERM} state, the call party is disconnected and the application can only gather information about the disconnected leg. From network session control, Idle_{TERM} and Releasing_{TERM} leg states are equivalent and can be labeled with the same label.

The SIP outgoing leg state model is synthesized from the suggested SIP session state model by refinement. The behavior refinement takes into account that adding or removing media is done by SIP request INVITE.

Let us denote by $T_{TERM} = (S_{TERM}, Act_{TERM}, \rightarrow, s_0)$ a labeled transition system representing the OSA application view on Terminating leg, where:

– S_{TERM} = { Idle_{TERM}, Active_{TERM} }

– Act_{TERM} = {routeReq, attachMedia, detachMedia, release, networkRelease}

– $\rightarrow$ = {Idle_{TERM} routeReq Active_{TERM},
Active_{TERM} attachMedia Active_{TERM},
Active_{TERM} detachMedia Active_{TERM},
Active_{TERM} release Idle_{TERM},
Active_{TERM} networkRelease Idle_{TERM}}

– s_0 = {Idle_{TERM}}.

We denote by $T_{SIPT} = (S_{SIPT}, Act_{SIPT}, \rightarrow_{SIPT}, s_0)$ a labeled transition system representing the state model of SIP terminating party regarded as outgoing leg where:

– S_{SIPT} = {Idle_{SIPT}, SessionProgress, WaitPRack, PRacked, WaitUpdate, Updated, Wait180, Alerting, WaitAck, Established, $\text{Wait200}_{\text{INVITE}}$, $\text{Wait200}_{\text{BYE}}$}

– Act_{SIPT} = {INVITE, 183, PRACK, 200_{PRACK}, UPDATE, 180, 200_{INVITE}, 200_{UPDATE}, ACK, $\text{BYE200}_{\text{BYE}}$ }

– $\rightarrow_{SIPT}$ = {Idle_{SIPT} INVITE SessionProgress,
SessionProgress 183 WaitPRack,
WaitPRack PRACK PRacked,
PRacked 200_{INVITE} WaitUpdate,
WaitUpdate UPDATE Updated,
Updated 200_{UPDATE} Wait180,
Wait180 180 Alerting,
Alerting 200_{INVITE} WaitAck,
WaitAck ACK Established,
Established INVITE $\text{Wait200}_{\text{INVITE}}$,
$\text{Wait200}_{\text{INVITE}}$ 200_{INVITE} Established
Established BYE $\text{Wait200}_{\text{BYE}}$,
$\text{Wait200}_{\text{BYE}}$, 200_{BYE} Idle_{SIPT}}

– s_0' = {Idle_{SIPT}}.

Proposition 3. The labeled transition systems T_{TERM} and T_{SIPT} are weakly bisimilar.

Proof. To prove the week bisimulation between T_{TERM} and T_{SIPT}we have to prove that there is a weak bisimulation relation between their states.

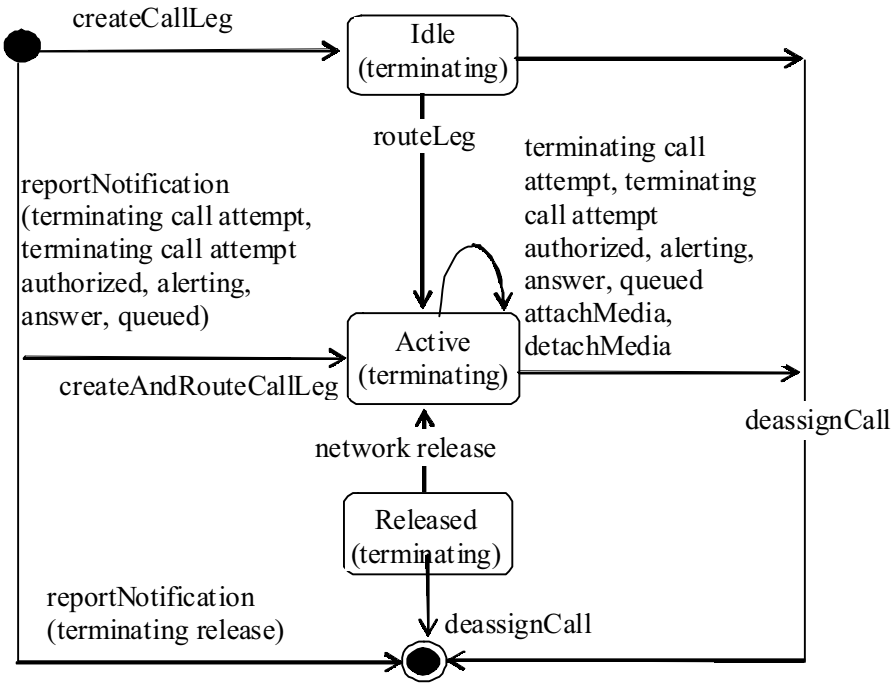

Figure 3. Terminating leg [3GRR TS 29.198-4-3]

Table 2. Bisimulation relations between OSA and SIP state models of a terminating leg

$s_1 \overset{a}{\Longrightarrow} s_2 \mid s_1, s_2 \in S_{TERM}$	$t_1 \overset{\hat{a}}{\Longrightarrow}' t_2 \mid t_1, t_2 \in S_{SIPT}$
$Idle_{TERM}$ routeReq $Active_{TERM}$	$Idle_{SIPT}$ INVITE SessionProgress, SessionProgress 183 WaitPRack, WaitPRack PRACK PRacked, PRacked 200_{PRACK} WaitUpdate, WaitUpdate UPDATE Updated, Updated 200_{UPDATE} Wait180, Wait180 180 Alerting, Alerting 200_{INVITE} WaitAck, WaitAck ACK Established
$Active_{TERM}$ attachMedia $Active_{TERM}$	Established INVITE $Wait200_{INVITE}$ $Wait200_{INVITE}$ 200_{INVITE} Established
$Active_{TERM}$ detachMedia $Active_{TERM}$	Established INVITE $Wait200_{INVITE}$ $Wait200_{INVITE}$ 200_{INVITE} Established
$Active_{TERM}$ release $Idle_{TERM}$	Established BYE $Wait200_{BYE}$ $Wait200_{BYE}$ 200_{BYE} $Idle_{SIPT}$
$Active_{TERM}$ networkRelease $Idle_{TERM}$	Established BYE $Wait200_{BYE}$ $Wait200_{BYE}$ 200_{BYE} $Idle_{SIPT}$

We define a binary relation U = { ($Idle_{TERM}$, $Idle_{SIPT}$), ($Active_{TERM}$, Established) }.

The proof that T_{TERM} and T_{SIPT} are weakly bisimilar follows from the relations between their states, shown in *table 2*.

B. Originating party state model

From the application point of view, the originating leg can be in one of five states as shown in *figure 4*. In the $Null_{ORIG}$ state, the originating leg object is undefined. In the $Initializing_{ORIG}$ state, the network verifies the authority of the originating party in order to establish a connection to the remote party. In the $Analyzing_{ORIG}$ state, the called party address is collected and additional address digits can be collected. Upon completion of address collection, the address is analyzed. In the $Active_{ORIG}$ state, a call leg connection to the calling party exists and originating mid-call events can be received. In the $Releasing_{ORIG}$ state, the connection to the call party is released, as requested by the network or application, reports are processed and sent to the application if requested, and the possible call leg information is collected. From the session control view, the $Null_{ORIG}$ и $Releasing_{ORIG}$ states are equivalent and can be labeled by the same label.

The suggested SIP incoming leg state model is refinement of the SIP session state model. The refinement takes into account that collecting of additional information from the user in IMS involves functionality of MRFC (Media Resource Function Controller). The Application has to establish connection to MRFC by SIP INVITE method. There are no standards for delivering the collected information to the application. There can be applied one of the following scenarios. The first one is when the MRFC collects the information and reports to the application by means of SIP BYE method requesting termination of the established session with the application. In the other scenario, the MRFC sends the collected information within the SIP INFO method and the application terminates the session by sending BYE request to the MRFC. Because of the similarity of the method for the state models and for brevity sake, we will consider just the first scenario. Further, in comparison with the terminating leg state model, for the originating leg it is possible to attach and detach media to the originating call leg not only in active state but during the connection establishment also. Modification of established SIP session is requested by INVITE method, while modification of SIP session during establishment is requested by UPDATE.

We present a method for modeling the behavior of OSA SCS in a role of Back-to-Back User Agent (B2BUA) where on receiving an inviting request the application requires additional information to be collected from the calling party (e.g. Credit card calling service). We use the following notations: SIP requests and responses related to the initial inviting request are

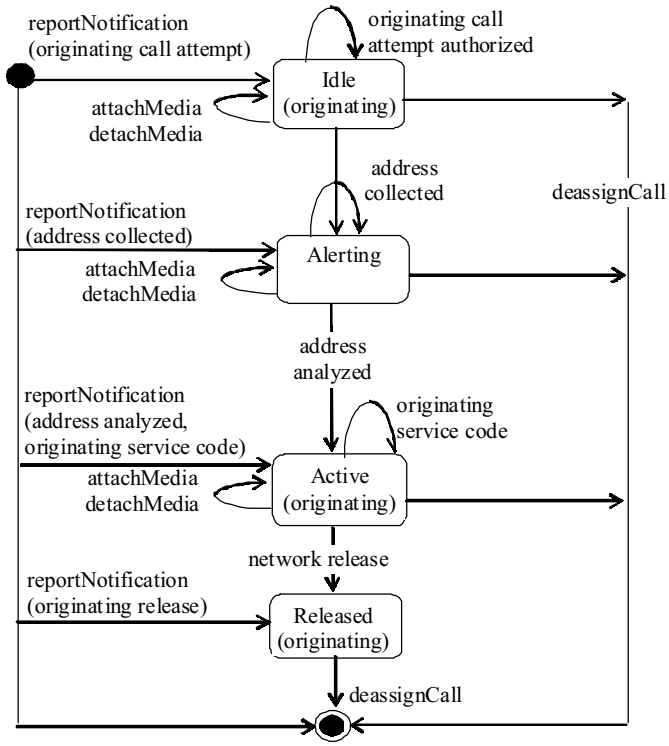

Figure 4. Originating Leg [3GRR TS 29.198-4-3]

Table 3. Bisimulation relations between OSA and SIP states of an originating leg

$s_1 \overset{a}{\Longrightarrow} s_2 \mid s_1, s_2 \in S_{ORIG}$	$t_1 \overset{\hat{a}}{\Longrightarrow}' t_2 \mid t_1, t_2 \in S_{SIPO}$
$Null_{ORIG}$ origCallAttempt $Initializing_{ORIG}$	$Idle_{SIPO}$ INVITE1 WaitApp1
$Initializing_{ORIG}$ origAttemptAuthorized $Initializing_{ORIG}$ $Initializing_{ORIG}$ attachMedia $Initializing_{ORIG}$, $Initializing_{ORIG}$ detachMedia $Initializing_{ORIG}$, $Initializing_{ORIG}$ addressCollected $Analyzing_{ORIG}$	WaitApp1 $INVITE_2$ $Wait200_{INVITE2}$, $Wait200_{INVITE2}$ $200_{INVITE2}$ SessionProgress1, SessionProgress1 183_1 WaitPRack1, WaitPRack1 $PRACK_1$ PRacked1, PRacked1 200_{PRACK1} WaitUpdate1, WaitUpdate1 $UPDATE_1$ Updated1, Updated1 $200_{UPDATE1}$ $Wait200_{INVITE1}$, $Wait200_{INVITE1}$ $200_{INVITE1}$ WaitAck2, WaitAck2 ACK_2 ConnectedToMediaResource, ConnectedToMediaResource BYE_2 $Wait200_{BYE2}$, $Wait200_{BYE2}$ 200_{BYE2} WaitApp2
$Analyzing_{ORIG}$ attachMedia $Analyzing_{ORIG}$, $Analyzing_{ORIG}$ detachMedia $Analyzing_{ORIG}$, $Analyzing_{ORIG}$ addressAnalyzed $Active_{ORIG}$,	WaitApp2 $INVITE_3$ SessionProgress3, SessionProgress3 183_3 WaitPRack3, WaitPRack3 $PRACK_3$ PRacked3, PRacked3 200_{PRACK3} WaitUpdate3, WaitUpdate3 $UPDATE_3$ Updated3, Updated3 $200_{UPDATE3}$ $Wait180_3$ $Wait180_3$ 180_3 Alerting3, Alerting3 $200_{INVITE3}$ WaitAck3, WaitAck3 ACK_3 Established3
$Active_{ORIG}$ networkRelease $Idle_{ORIG}$	Established3 BYE_3 $Wait200_{BYE3}$, $Wait200_{BYE3}$, 200_{BYE3} $Idle_{SIPO}$

suffixed by 1, SIP requests and responses related to the session with MRFC are suffixed by 2, and SIP requests and responses related to the new session initiation request are suffixed by 3.

Let us denote by T_{ORIG}= (S_{ORIG}, Act_{ORIG}, $\rightarrow$, s_0) labeled transition system representing the view of OSA application on the originating leg states where:

- S_{ORIG} = {$Null_{ORIG}$, $Initializing_{ORIG}$, $Analyzing_{ORIG}$, $Active_{ORIG}$}
- Act_{ORIG} = {origCallAttempt, origAttemptAuthorized, addressCollected, addressAnalyzed, attachMedia, detachMedia, networkRelease}
- $\rightarrow$ = { $Null_{ORIG}$ origCallAttempt $Initializing_{ORIG}$, $Initializing_{ORIG}$ origAttemptAuthorized $Initializing_{ORIG}$, $Initializing_{ORIG}$ attachMedia $Initializing_{ORIG}$, $Initializing_{ORIG}$ detachMedia $Initializing_{ORIG}$, $Initializing_{ORIG}$ addressCollected $Analyzing_{ORIG}$, $Analyzing_{ORIG}$ attachMedia $Analyzing_{ORIG}$, $Analyzing_{ORIG}$ detachMedia $Analyzing_{ORIG}$, $Analyzing_{ORIG}$ addressAnalyzed $Active_{ORIG}$, $Initializing_{ORIG}$ release $Idle_{ORIG}$, $Active_{ORIG}$ attachMedia $Active_{ORIG}$, $Active_{ORIG}$ detachMedia $Active_{ORIG}$, $Active_{ORIG}$ networkRelease $Idle_{ORIG}$}
- s_0 = {$Idle_{ORIG}$}.

By T_{SIPO} = (S_{SIPO}, Act_{SIPO}, $\rightarrow_{SIPO}$, s_0') we denote a label transition system representing the SIP incoming leg state model where:

- S_{SIPO}={$Idle_{SIPO}$,WaitApp1,$Wait200_{INVITE2}$, SessionProgress1, WaitPRack1, PRacked1, WaitUpdate1, Updated1, $Wait200_{INVITE1}$, WaitAck2, ConnectedToMediaResource, $Wait200_{BYE2}$, WaitApp2, SessionProgress3, WaitPRack3, PRacked3, WaitUpdate3, Updated3, $Wait180_3$, Alerting3, WaitAck3, Established3, $Wait200_{BYE3}$}
- Act_{SIPO} = {$INVITE_1$, $INVITE_2$, $200_{INVITE2}$, 183_1, $PRACK_1$, 200_{PRACK1}, $UPDATE_1$, $200_{UPDATE1}$, $200_{INVITE1}$, ACK_2, BYE_2, 200_{BYE2}, $INVITE_3$, 183_3, $PRACK_3$, 200_{PRACK3}, $UPDATE_3$, $200_{INVITE3}$,180_3, $200_{INVITE3}$, ACK_3, BYE_3, 200_{BYE3} }
- $\rightarrow_{SIPO}$ = {$Idle_{SIPO}$ $INVITE_1$ WaitApp1, WaitApp1 $INVITE_2$ $Wait200_{INVITE2}$, $Wait200_{INVITE2}$ $200_{INVITE2}$ SessionProgress1, SessionProgress1 183_1 WaitPRack1, WaitPRack1 $PRACK_1$ PRacked1, PRacked1 200_{PRACK1} WaitUpdate1, WaitUpdate1 $UPDATE_1$ Updated1, Updated1 $200_{UPDATE1}$ $Wait200_{INVITE1}$, $Wait200_{INVITE1}$ $200_{INVITE1}$ WaitAck2, WaitAck2 ACK_2 ConnectedToMediaResource, ConnectedToMediaResource BYE_2 $Wait200_{BYE2}$, $Wait200_{BYE2}$ 200_{BYE2} WaitApp2, WaitApp2 $INVITE_3$ SessionProgress3, SessionProgress3 183_3 WaitPRack3, WaitPRack3 $PRACK_3$ PRacked3, PRacked3 200_{PRACK3} WaitUpdate3,

WaitUpdate3 $UPDATE_3$ Updated3,
Updated3 $200_{UPDATE3}$ $Wait180_3$,
$Wait180_3$ 180_3 Alerting3,
Alerting3 $200_{INVITE3}$ WaitAck3,
WaitAck3 ACK_3 Established3,
Established3 BYE_3 $Wait200_{BYE3}$,
$Wait200_{BYE3}$, 200_{BYE3} $Idle_{SIPO}$ }

– s_0' = {$Idle_{SIPO}$}.

Proposition 4. The labeled transition systems T_{ORIG} and T_{SIPO} are weakly bisimilar.

Proof. The weak bisimilarity between T_{ORIG} и T_{SIPO} will be proved by identification of bisimulation relation between their states.

We denote by U a binary relation where U = { ($Null_{ORIG}$, $Idle_{SIPO}$), ($Initialyzing_{ORIG}$, WaitApp1), ($Analyzing_{ORIG}$, WaitApp2, $Active_{ORIG}$, Established3) }.

The statement that U is a bisimulation relation follows from the relation shown in *table3*.

The considered SIP incoming leg state model is simplified. It can be extended by adding the mapping of the IpCallLeg release method by which the application requires to disconnect the call onto SIP BYE method, if the release method is invoked in the $Active_{ORIG}$ state or onto SIP CANCEL method if the release method is invoked while being in the $Initializing_{ORIG}$ or $Analyzing_{ORIG}$ state. Transitions related to the SIP 3xx, 4xx, 5xx и 6xx responses have to be further refined for which the application is notified by eventReportRes method.

V. Use Case for OSA Application Control on SIP Conferencing

To illustrate the behavior of the OSA SCS running both OSA and SIP state machines we provide a use case of OSA application control on SIP conferencing.

The SIP is used to establish, modify and terminate multimedia conferences. In [13], a framework for SIP conference control is defined. High level requirement to SIP conferencing are identified in [14].

The OSA specifications define several service capabilities features for call control. The Conference Call Control SCF [12] inherits the interface methods of Multiparty Call Control SCF adding specialized capabilities for multimedia conferences.

The suggested SIP session state models can be applied to SIP conference call states adding transitions in the SIP incoming and outgoing leg state models related to subscription for and notification of conference events.

The possible application scenario is a prearranged conference for a specified time period. During this timeslot parties can call in to the conference by dialling a special number. This scenario describes a case when a participant knows the conference URI and tries to get connected to the conference. The OSA application is responsible for the participant's authentication and authorization.

Figure 5 shows a call flow for joining to prearranged conference using the conference URI. As a precondition the

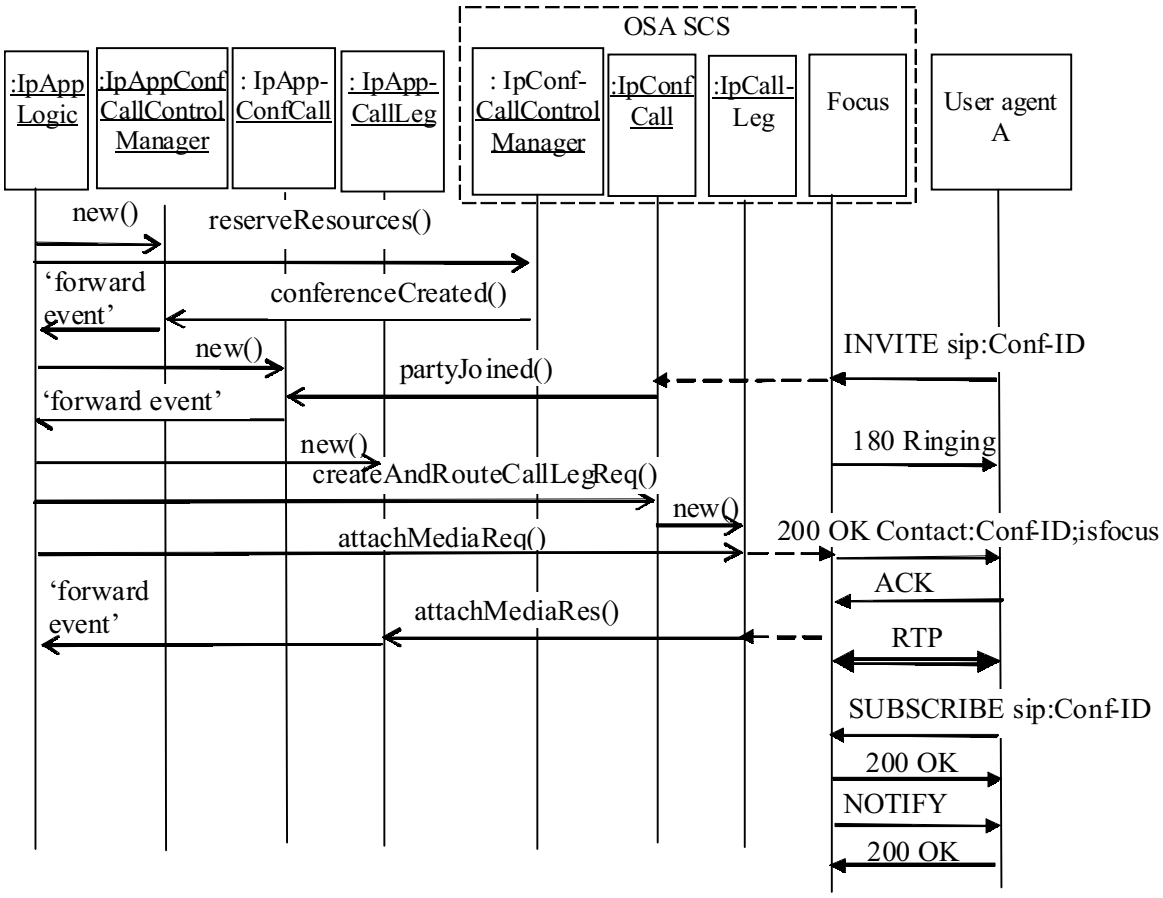

Figure 5. Application initiated joining to prearranged conference

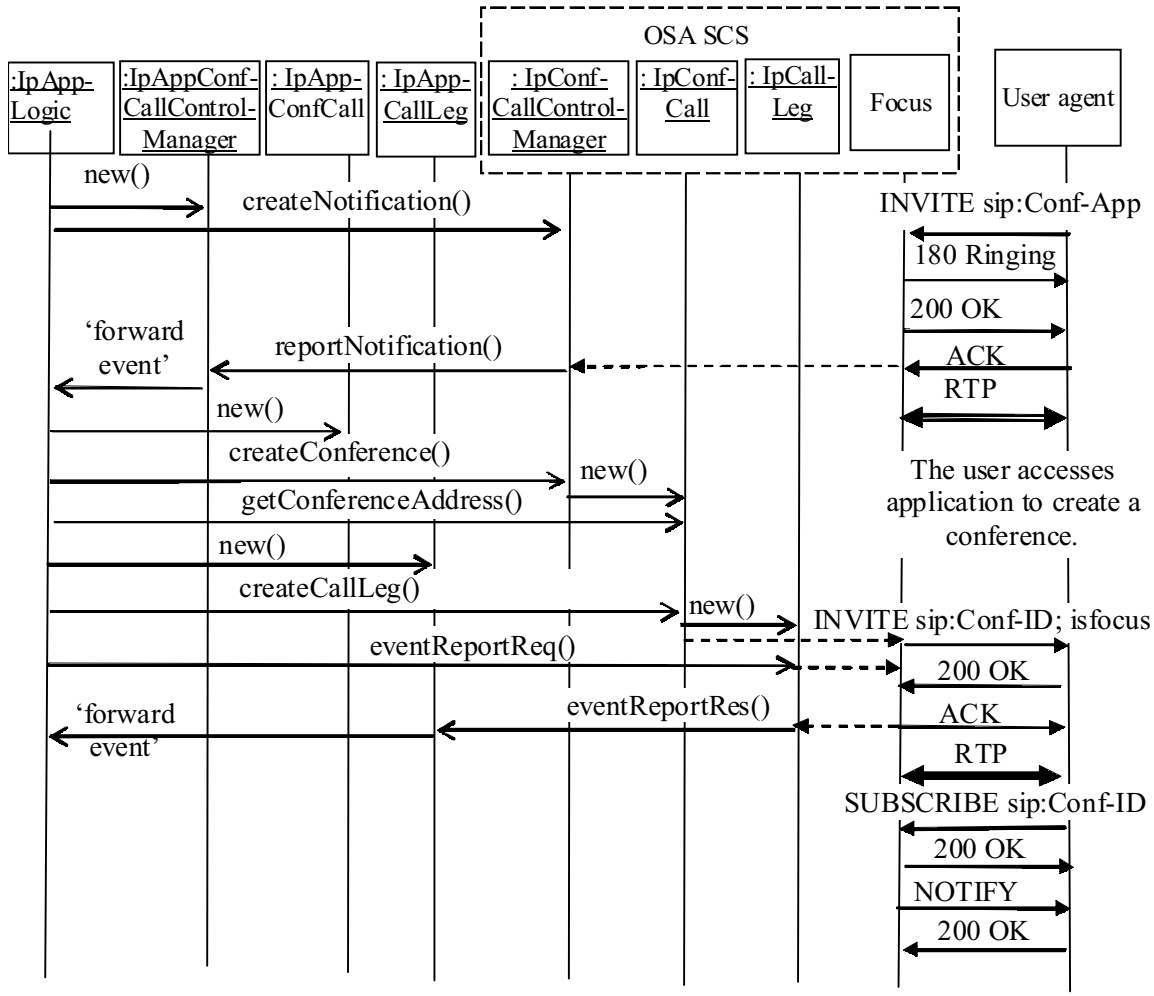

Figure 6. Manual creation of a conference by dialling in to a conferencing application

application has reserved conference resources for some time in the future (i.e. the conference URI should be reserved, before its actual use). The application also has registered its interest in the creation of the conference. The focus is activated when the first party joins the conference or after the specified start time.

The focus behavior follows the proposed SIP session state model, while the user agents invited to participate in the conference follow the behavior of terminating parties. The application treats the invited user agents as terminating legs.

The conference is created at the specified start time and the application is informed by invoking its method conferenceCreated(). A conference-aware user agent issues an INVITE message, containing the conference's URI as a value of the Request-URI header. The application is notified that a new party joined the conference by invoking its method partyJoined(). The application decides that the party is allowed to join the conference and invokes the attachMediaReq() method to add the party. The focus sets 'isfocus' parameter in the 'Contact' header of the response. Alternatively, the party could be rejected with a releaseCallLeg. Otherwise, the joined user agent subscribes to the conference events.

The so-called 'manual' creation of a conference is conducted by sending an INVITE to a conference server application. The necessity of such an application is because the setup of a conference requires a set of additional parameters. After establishing a dialog and receiving the parameters using non-SIP means, the creation of the conference and respective focus is possible.

Figure 6 shows a call flow for manual creation of a conference by dialling to a conference application.

The application has registered its interest about events regarding dialling conference server number. When a user sends an INVITE to a conference server the application is notified. By the use of non-SIP means for user interaction, the application collects additional information and 'creates' the focus. A new conference call object is created. If the user is going to be conference participant then the application adds the participant. The focus re-invites the user by 'Contact' header with 'isfocus' parameter and after acknowledgment the application is notified. In this scenario, the focus behavior follows the proposed SIP session state model, while the user agent, calling to the conference number is regarded by the application and focus as terminating leg.

VI. Conclusion

The network interface of the OSA SCS is not standardized and is regarded as an implementation detail. There is a con-

straint for any implementations for single point of contact. Instead of coordinating the connections in circuit switched and packed switched networks, the OSA SCS governs the interface to the underlying network. Bearer and media connections rely on connections made by the call session signaling, e.g. SIP signaling. Interfacing between application programming interfaces requests and responses, and SIP signaling, the OSA SCS needs to care for both state machines – one for the interface objects and another for the SIP session.

We suggest a method for modeling the SIP session state in an OSA SCS which supports interfaces for multiparty call control. While the SIP session state model provides a high level abstraction for handling multiparty calls, differences in controlling the states of the incoming and outgoing legs have to be considered. More refined session model is achieved when consider the different roles played by application in controlling SIP sessions. By presenting the models in a formal way as label transition systems, the behavior equivalence is proved using the concept of bisimulation.

Following the same approach, session state models for CAP signaling might be designed. For example, following the mapping of OSA Generic Call Control interface methods onto CAP methods, it can be proved formally that depending on the application role, CAMEL originating and terminating basic call state models might be used. Further, OSA Data Session Control SCF, which provides capabilities for control over packet sessions in mobile networks, can be used to design data session model corresponding to the PDP context state model.

The method is useful in the design and implementation of OSA Service Capability Server. Because of the OSA Service Capability Server complexity, the model-based testing techniques assist in its systematization. By starting from a formal model, test cases can be derived automatically in order to prove the conformance of implementations with respect to their specifications. Automation of some parts of the testing activity, using models and formal methods is a way to improve the quality and to reduce the cost of design.

Acknowledgement

The work is conducted under the grant of the Project DO 02-135/2008 Research on *Cross Layer Optimization of Telecommunication Resource Allocation*, funded by Bulgarian Ministry of Education and Science.

References

1. 3GPP TS 23.218 IP Multimedia (IM) Session Handling; IM Call Model, 9.0.0, 2009.

2. Chlamtac, Imrich, Hsin-Yi Lee, Yi-Bing Lin, Meng-Hsun Tsai. An OSA Service Capability Server for Mobile Services. – *International Journal of Pervasive Computing and Communications*, 4, 2008, issue 3, 268-278.

3. Carlín, J. M. E. Conformance Evaluation of the Session Control Mechanisms of an IMS Testbed. – Proceedings of Communication Systems and Networks, AsiaCSN, 2008, paper 607-133.

4. Gouya, A., N. Crespi, E. Bertin, L. Oueslati. Managing Service Capability and Service Feature Interactions in the IMS of UMTS. Proceedings of International Conference on Networking and Services (ICNS'06), 2006, 50-55.

5. Walker, Stuart. Providing IMS Services to Legacy Network Endpoints. <http://www.iec.org/newsletter/august07_1/analyst_corner.pdf>, 2009.

6. J´eron, T. Symbolic Model-based Test Selection. Electronic Notes in Theoretical Computer Science, 240, 2009, 167-184

7. Aichernig, B., M. Weiglhofer, F. Wotawa. Improving Fault-based Conformance Testing. Electronic Notes in Theoretical Computer Science, 220, 2008, 63-77

8. Chena, Xiao Jun, Rocco De Nicola. Algebraic Characterizations of Trace and Decorated Trace Equivalences over Tree-like Structures. Theoretical Computer Science, 2001, 337-361.

9. Panangaden, Prakash. Notes on Labelled Transition Systems and Bisimulation, <http://www.cs.mcgill.ca/~prakash/Courses/comp330/Notes/lts09.pdf>, 2004.

10. 3GPP TS 29.198-4-3, Open Service Access (OSA); Application Programming Interface (API) Part 4: Call Control Sub-part 3:Multi-party Call Control Service Capability Feature; 9.0.0, 2009.

11. 3GPP TR 29.998-04-4, Open Service Access (OSA); Mapping for Open Service Access; Part 4: Call Control Service Mapping; Subpart 4: Multiparty Call Control ISC, 9.0.0, 2009.

12. 3GPP TS 29.198-04-05 Open Service Access (OSA); Application Programming Interface (API) Part 4: Call Control Sub-part 5: Conference Call Control Service Capability Feature; 9.0.0. 2009.

13. Rosenberg, J. A Framework for Conferencing with the Session Initiation Protocol (SIP). RFC 4353, 2006.

14. Levin, O., R. Even. High-Level Requirements for Tightly Coupled SIP Conferencing. RFC 4245, 2005.

Manuscript received on 09.02.2010

Evelina Pencheva *received M.S. degree in mathematics at the University of Sofia, Bulgaria. She obtained PhD degree in Telecommunications from Technical University of Sofia. She is Associate Professor at Faculty of Telecommunications, Technical University of Sofia. Her academic activity and experience is in lecture courses on telecommunication networks and service technologies. Her interests include next generation mobile applications and middleware platforms.*

Contacts:
Technical University of Sofia
tel: +359 2 965 3695
e-mail: enp@tu-sofia.bg

Ivaylo Atanasov *received M.S. degree in electronics at the Technical University of Sofia, Bulgaria. He obtained PhD degree in Telecommunication networks. His current position is Associate Professor at Faculty of Telecommunications, Technical University of Sofia. The academic experience is in courses on object-oriented programming, mobile networks and service creation technologies. The main research focus is set on the field of open service platforms for next generation networks.*

Contacts:
Technical University of Sofia
tel: +359 2 965 2050
e-mail: iia@tu-sofia.bg