Third Party User Interaction Control in SIP Networks

Ivaylo Atanasov, Evelina Pencheva

Third Party User Interaction Control in SIP Networks

Ivaylo Atanasov, Evelina Pencheva

Technical University of Sofia, Faculty of Telecommunications
8 Kliment Ohridski blvd., 1000 Sofia, Bulgaria
{iia,enp}@tu-sofia.bg

Abstract. A lot of attractive applications in addition to manipulation of session related signaling involve specific processing at media level such as playing media, prompting and collecting media from the user, mixing media streams etc. One of the ways of provisioning applications in managed all IP-based multimedia networks, is based on Open Service Access (OSA) service platform. The paper investigates the capabilities for OSA third party control on user interactions in multimedia networks where the session management is based on Session Initiation Protocol (SIP). The focus is on the interoperability between OSA application control on session-related user interactions and media services in SIP networks. OSA user interaction interfaces are mapped onto SIP signaling. The behavior of the OSA gateway is modeled by synchronization of application view on user interaction call and SIP session involving media resources. A formal approach to functional verification of OSA gateway is proposed.

Key words: Open Service Access, Media services, Interface to protocol mapping, Formal testing of functional behavior

1 Introduction

Media services refer to different type of functions such as playing media, prompting and collecting media from the user, and mixing media. Many of existing multimedia applications use media services. In a managed all Internet Protocol-based network, session control relies on Session Initiation Protocol (SIP) [1]. SIP signaling is used to control media resource functions which provide media services to users.

Multimedia applications may be provided by three types of application servers [2].

SIP-based application servers host a wide range of value-added multimedia services. Along with the services like presence and availability, messaging and conferencing, SIP-based application server may be programmed to provide media services controlled by VoiceXML scripts [3], [4]. A mechanism for providing an interoperable interface between SIP-based application servers and media servers is defined in [5], [6].

The second alternative for service provisioning is Customized Applications for Mobile Network Enhanced Logic (CAMEL) Service Environment which offers

capabilities for supporting legacy services such as call control, user interaction, user status etc. The interworking between CAMEL services and SIP session control is defined in [7].

Open Service Architecture (OSA) allows third party access to communication functions in a network neutral way. Using OSA Application Programming Interfaces (APIs), application developers can create attractive applications without specific knowledge about underlying network technology and control protocols. Interoperability between OSA applications and specific network functions requires special type of application server called OSA gateway. The OSA gateway is responsible for translation of OSA interface method invocations into control protocol messages and vice versa.

The OSA User Interaction (UI) service provides API for call-related and call-unrelated user interactions [8]. The user interaction supports sending information or sending and collecting information. The mappings of OSA UI API onto CAMEL Application Part protocol and Short Message Service are defined in [9] and [10] respectively. No mapping of OSA UI API onto SIP signaling is defined. As SIP is considered to be a key control protocol in all IP-based multimedia networks, the mapping between OSA UI API and SIP would allow third party control on both session management and media services.

In this paper we investigate the interoperability between OSA application control on user interactions and SIP-based media services. We provide mapping of OSA UI API onto SIP signaling following [11]. As the OSA application view on user interactions has to be synchronized with the SIP session state in media resources, we suggest a formal approach to specification of functional behavior of OSA gateway.

The paper is organized as follows. In Section 2, we discuss aspects of OSA deployment in managed all IP-based multimedia networks and present in brief the OSA User Interaction service. In Section 3, we define an interoperability mapping between OSA UI API and SIP protocol. A formal description of OSA gateway behavior is provided in Section 4, where a SIP session state model representing the states of media resources is proposed and the behavioral equivalence between SIP session handling and OSA application user interaction handling is proved. Before concluding the paper we present an example of OSA application control on media services.

2 OSA Application Control in Managed All-IP Multimedia Networks

Internet Protocol Multimedia Subsystem (IMS) is service control architecture intended to provide all types of multimedia services based on IP connectivity [12]. In the IMS control architecture, Application Servers run applications some of which may reside in a third party network as shown in Fig. 1. The OSA gateway provides interoperability between third party control and network functions. The OSA gateway communicates with Serving CSCF (S-CSCF) which is responsible

for user registration and session management which rely on SIP signaling. Media Resource Function Controller (MRFC) and Media Resource Function Processor (MRFP) together provide mechanisms for media services such as conferencing, announcements to users or bearer transcoding in the IMS architecture. The MRFC handles SIP communication to and from the S-CSCF and controls the media resources of MRFP using H.248 protocol. The MRFP provides media resources requested and instructed by the MRFC. The Home Subscriber Server (HSS) is a database which stores the user profiles. The access to user data in the HSS is based on Diameter signaling.

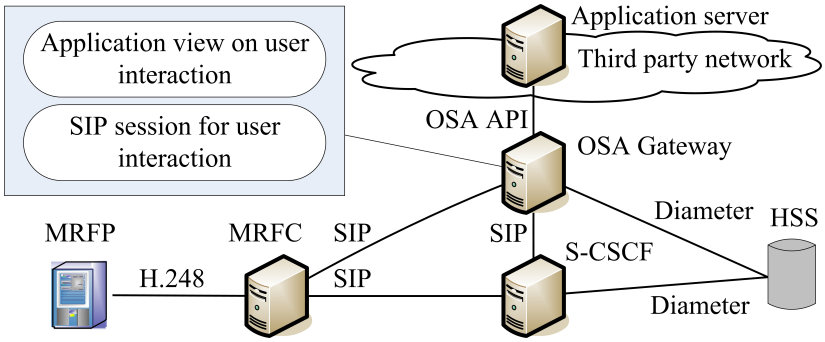

Fig. 1. Open access to media services in IMS

The OSA User Interaction service distinguishes between two levels of user interactions. Generic user interaction supports sending information or sending and collecting information from users. Call-related user interaction allows interactions with users engage in call.

In cases of OSA application control on user interactions in IMS, the OSA gateway is responsible for the translation of OSA methods into SIP messages and vice versa. The OSA gateway provides interoperability between the state machines representing the application view on user interaction and the SIP session with MRFC. In the next section, we define an interoperability mapping between OSA User Interaction interface methods and SIP messages.

3 Mapping of OSA User Interaction Interfaces onto SIP Protocol

The OSA User Interaction API provides the third party applications with access to the functions such as playing media, prompting and collecting media from the user via the OSA Interface Class methods. At the OSA gateway, the OSA Interface Class methods need to be mapped, or translated, onto the relevant SIP methods. The present paper is not exhaustive in covering all the mappings that can be expected. In particular, only general cases of normal operations

are covered and exception scenarios are not considered. The elaborate mapping requires also parameters mapping which is beyond the scope of the paper.

The OSA *createUI* and *createUICall* methods are used to create a new User Interaction object for non-call related and call related purposes respectively. The invocation of these methods results in SIP session establishment with MRFC as shown in Fig. 2.

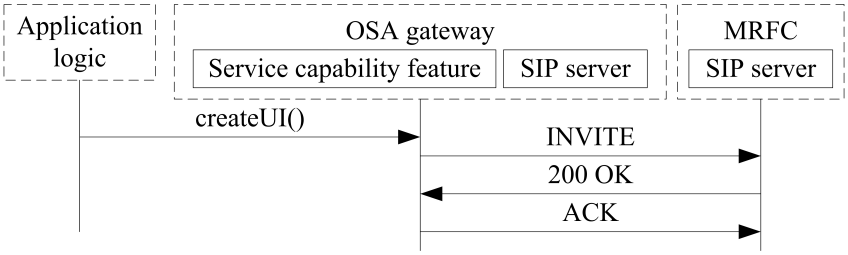

Fig. 2. Creation of OSA UI object and session initiation with MRFC

The OSA asynchronous *sendInfoReq* method sends information to the user. This information can be sent by SIP INFO request which controls media resources within the established session with MRFC as shown in Fig. 3. The INFO request may be used for initial or subsequent *sendInfoReq* method. The information to be sent to the user is transferred in the body of INFO request, for example as an XML script.

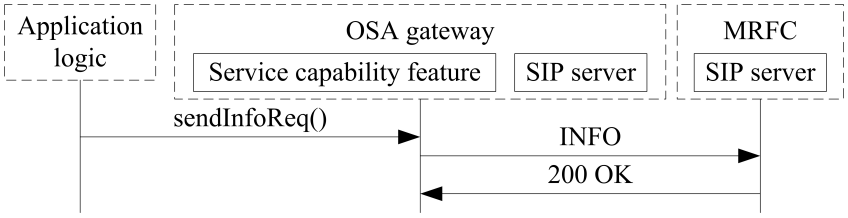

Fig. 3. Mapping of *sendInfoReq* method onto INFO message

The OSA asynchronous *sendInfoRes* method informs the application about the start or the completion of a *sendInfoReq* method. This response is called only if the application has requested a response and can be mapped onto SIP INFO request as shown in Fig. 4. The user interaction report is transferred in the INFO request body.

The SIP INFO request can be used to indicate that the request to send information was unsuccessful which is reported to the application by invoking of OSA *sendInfoErr* method. The user interaction error code is transferred in the body of the INFO message.

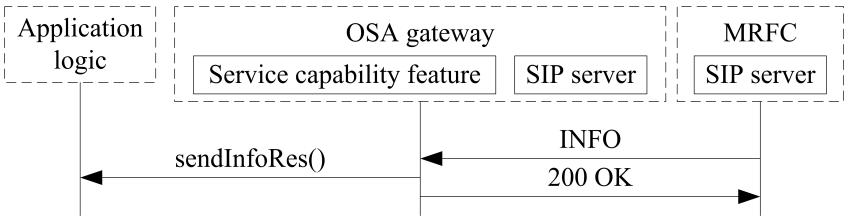

Fig. 4. Mapping of *sendInfoRes* method onto INFO message

The OSA $sendInfoAndCollectReq$ method that plays an announcement or sends other information to the user and collects some information from the user can be mapped onto SIP INFO message as shown in Fig. 5. The information collected that has to be returned to the application using the $sendInfoAndCollectRes$ method is sent by INFO message also.

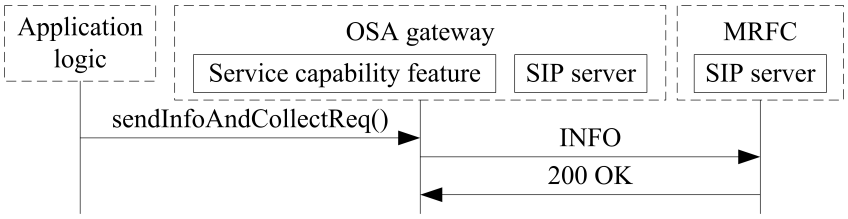

Fig. 5. Mapping of *sendInfoAndCollectReq* method onto INFO message

The INFO message is used to transfer the indication that the request to send information and collect a response was unsuccessful which is reported to the application by $sendInfoAndCollectErr$ method as shown in Fig. 6.

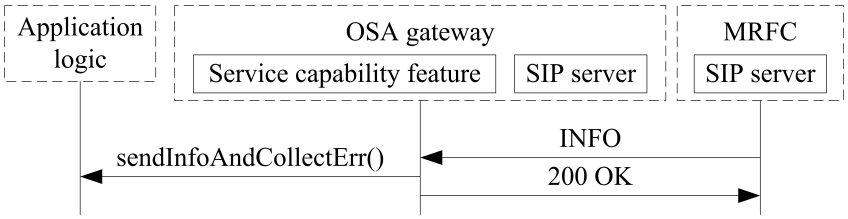

Fig. 6. Mapping of *sendInfoAndCollectErr* method onto INFO message

The OSA *release* method requests that the relationship between the application and the user interaction object be released. It causes the release of the used user interaction resources and interrupts any ongoing user interaction. The result is termination of the SIP session with the MRFC as shown in Fig. 7.

The OSA $createNotification$, $changeNotification$, $destroyNotification$, $enableNotifications$, and $disableNotifications$ methods of the IpUIManager

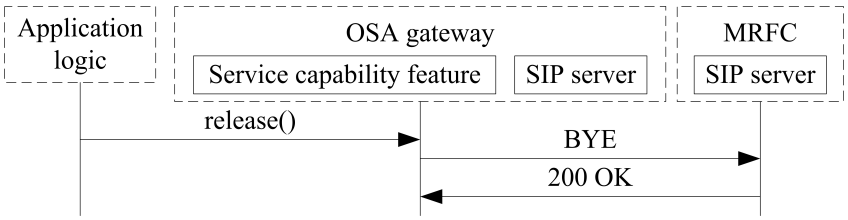

Fig. 7. The application initiated release of user interaction resources

are used to create, change, destroy, enable, and disable notifications for user initiated interactions.

Fig. 8 shows the call flow for *createNotification* method. The OSA gateway requests the S-CSCF to observe for certain SIP events to be notified to the application. Initial filtering information will be uploaded to the HSS and from here to the S-CSCF, e.g. when the user gets registered. User-related data in the HSS are updated by Diameter signalling.

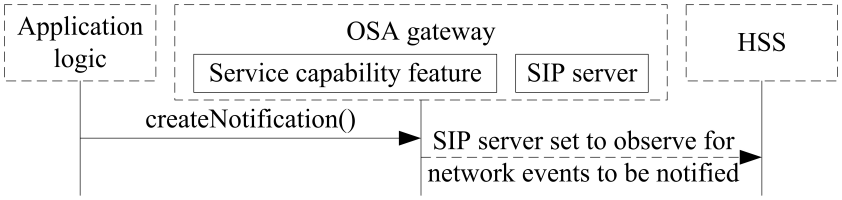

Fig. 8. The application subscribes for notifications related to user session

The OSA *reportEventNotification* method notifies the application of an occurred network event which matches the criteria installed by the *createNotification* method. Fig. 9 and Fig. 10 show the call flows for *reportEventNotification* triggered by SIP requests and SIP answers respectively.

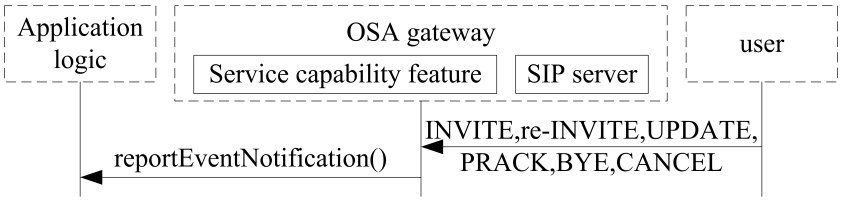

Fig. 9. Notification triggered by SIP requests

The OSA *userInteractionAborted* method indicates to the application that the User Interaction service instance has terminated or closed abnormally (e.g. a link failure to the MRFP). This event is reported by MRFC using SIP BYE message.

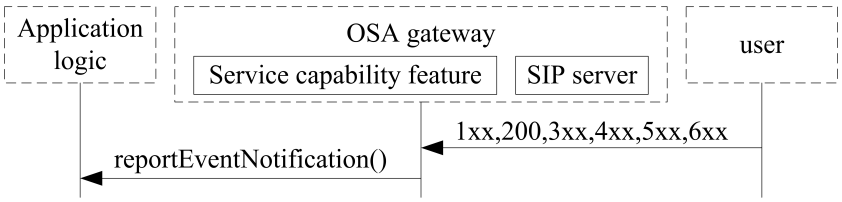

Fig. 10. Notification triggered by SIP responses

4 Formal Specification of Interoperable User Interaction Model

The formal specification of finite state machines as Labeled Transition Systems allows proving the behavioral equivalence and hence the interoperability of OSA user interaction control and IMS media service control. This may be used for automatic generation of test cases during the OSA gateway verification.

4.1 Labeled Transition Systems and Behavioral Equivalence

To prove behavioral equivalence between state machines formally, the notion of *Labeled Transition Systems* is used [13].

Definition 1. *A Labeled Transition System (LTS) is a quadruple* $(S, Act, \rightarrow, s_0)$*, where* S *is 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.*

We will use the following notations:
$s \xrightarrow{a} s'$ stands for the transition (s, a, s');
$s \xrightarrow{a}$ means that $\exists s' : s \xrightarrow{a} s'$;
$s \stackrel{\mu}{\Rightarrow} s_n$ where $\mu = a_1, a_2, ..., a_n : \exists s, s_1, ..., s_n$ s.t. $s \xrightarrow{a_1} s_1 ... \xrightarrow{a_n} s_n$;
$s \stackrel{\mu}{\Rightarrow}$ means that $\exists s' : s \stackrel{\mu}{\Rightarrow} s'$;
$\stackrel{\hat{\mu}}{\Rightarrow}$ means that $\Rightarrow$ if $\mu \equiv \tau$ or $\stackrel{\mu}{\Rightarrow}$ otherwise
where τ is one or more internal (invisible) actions. More detailed notation description can be found in [13].

The concept of bisimulation [14] is used to prove that two LTS expose equivalent behavior. 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 2. *[13] 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 that if* $s_1 U t_1 : s_1 \subseteq S, t_1 \subseteq S'$ *then* $\forall a \in Act$*:*

$s_1 \stackrel{a}{\Rightarrow} s_2$ *implies* $\exists t_2 : t_1 \stackrel{\hat{a}}{\Rightarrow}' t_2$ *and* $s_2 U t_2$*;*
$t_1 \stackrel{a}{\Rightarrow}' t_2$ *implies* $\exists s_2 : s_1 \stackrel{\hat{a}}{\Rightarrow} s_2$ *and* $s_2 U t_2$*.*

4.2 Formal Description of OSA User Interaction

The application view on UI object is defined in [8]. The behavior of the UI object can be described by finite state machine. In *Null* state, the UI object does not exist. The UI object is created when the *createUI* method is invoked or a network event is reported by *reportEventNotification* method. In *Active* state, the UI object is available for requesting messages which have to be sent to the network. Both *sendInfoAndCollectReq* and *sendInfoReq* methods have a parameter indicating whether it is the final request and whether the UI object has to be released after the information has been presented to the user. In *Active* state, in case a fault is detected on the user interaction, an error is reported on all outstanding requests. A transition to *ReleasePending* state is made when the application has indicated that after a certain message no further messages need to be sent to the end-user. There are, however, still a number of messages that are not yet completed. When the last message is sent or when the last user interaction has been obtained, the UI object is destroyed. In *Finished* state, the user interaction has ended. The application can only release the UI object. A simplified state transition diagram for UI object is shown in Fig. 11.

By $T_{AppUI} = (S_{AppUI}, Act_{AppUI}, \rightarrow_{AppUI}, s'_0)$ we denote a LTS representing the OSA application view on UI object. Final and non-final attributes are shortly marked as X[f] and X[!f] respectively. Then:

S_{AppUI} = { *Null, Active, ReleasePending, Finished* };

Act_{AppUI} = { *createUI, sendInfoRes[!f], reportEventNotification, userInteractionAborted, sendInfoReq[!f], sendInfoErr[f], sendInfoAndCollectReq[f], sendInfoErr[!f], sendInfoRes[f], sendInfoAndCollectReq[!f], sendInfoReq[f], release, sendInfoAndCollectRes[f], sendInfoAndCollectRes[!f], sendInfoAndCollectErr[f], sendInfoAndCollectErr[!f]* };

$\rightarrow_{AppUI}$ = {*Null createUI Active, Null reportEventNotification Active, Active sendInfoReq[!f] Active, Active sendInfoRes[!f] Active, Active sendInfoAndCollectReq[!f] Active, Active sendInfoAndCollectRes[!f] Active, Active sendInfoErr[!f] Active, Active sendInfoAndCollectErr[!f] Active, Active release Null, Active sendInfoReq[f] ReleasePending, Active sendInfoRes[!f] ReleasePending, ReleasePending sendInfoErr[f] Active, ReleasePending sendInfoErr[!f] ReleasePending, ReleasePending sendInfoRes[f] Finished, ReleasePending userInteractionAborted Finished, ReleasePending release Null, Finished release Null, Active sendInfoAndCollectReq[f] ReleasePending, ReleasePending sendInfoAndCollectErr[f] Active, ReleasePending sendInfoAndCollectRes[!f] ReleasePending, ReleasePending sendInfoAndCollectErr[!f] ReleasePending, ReleasePending sendInfoAndCollectRes[f] Finished, Active sendInfoReq[f] Finished, Active userInteractionAborted Finished* };

s'_0 = { *Null* }.

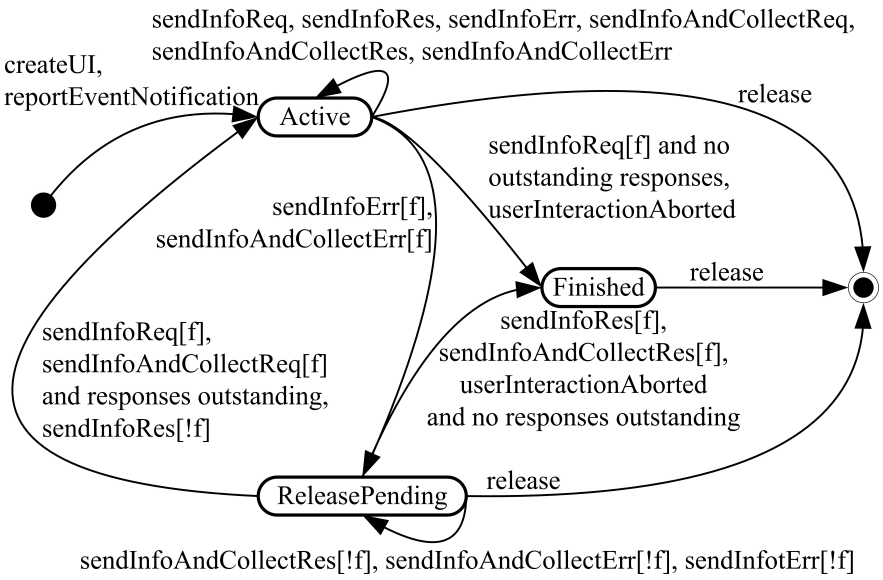

Fig. 11. OSA application view on the UI object

4.3 Formal Description of SIP Session with MRFC

We formalize the SIP session with MRFC by $T_{SIP} = (S_{SIP}, Act_{SIP}, \rightarrow_{SIP}, s_0)$. It is an LTS which represents a simplified SIP session state machine where:

S_{SIP} $= \{ Idle, Wait200_{INVITE}, Established, Wait200_{INFO}, Wait200_{BYE} \}$;

$Act_{SIP} = \{ INVITE, 200_{INVITE}, INFO, 200_{INFO}, BYE, 200_{BYE} \}$;

$\rightarrow_{SIP}$ $= \{ Idle\ INVITE\ Wait200_{INVITE},$
$Wait200_{INVITE}\ 200_{INVITE}\ Established,$
$Established\ INFO\ Wait200_{INFO},$
$Wait200_{INFO}\ 200_{INFO}\ Established,$
$Established\ BYE\ Wait200_{BYE},$
$Wait200_{BYE}\ 200_{BYE}\ Idle \}$;

s_0 $= \{ Null \}$.

4.4 Interoperability between OSA User Interactions and IMS Media Services

To prove the interoperability between user interaction models in OSA and IMS we have to prove that the state machine representing the OSA user interactions and the SIP state machine expose equivalent behavior. The behavioral equivalence is proved using the concept of weak bisimilarity.

Proposition 1. *The labeled transition systems T_{AppUI} and T_{SIP} are weakly bisimilar.*

Proof. To prove that bisimulation relation exists between two labeled transition systems, it has to be proved that there is a bisimulation relation between their states. By U it is denoted a relation between the states of T_{AppUI} and T_{SIP} where $U = \{(Null, Idle), (Active, Established)\}$. Table 1 presents the bisimulation relation between the states of T_{AppUI} and T_{SIP} which satisfies the Definition 2. The mapping between the OSA User Interaction interface methods and SIP messages defined in Section 3 shows the actions' similarity. Based on the bisimulation relation between the states of T_{AppUI} and T_{SIP} it is proved that both systems expose equivalent behavior.

5 An Example of OSA Application Control on User Interaction in IMS

Let us consider an example application that sends greetings on occasion (e.g. on birthdays). The sequence diagram in Fig. 12 shows a 'greeting message', in the form of an announcement, being delivered to a user as a result of a trigger from an application. Typically, the application would be set to trigger at certain time, however, the application could also trigger on events. The application initiates a call to the user (steps 1-4), the call is setup in the network (steps 5-7) and the application is informed about the result of call setup (steps 8-9). Then the application logic determines to play an announcement to the user and creates a user interaction call (steps 10-11), the OSA gateway initiates a SIP session to the MRFC (steps 11-17). The application sends the greeting information to be sent to the user (step 18) and the OSA gateway sends instructions to the MRFC (steps 19-22). When the greeting is played the MRFC reports (steps 23-26) and the report is forwarded to the application (steps 27-28). At last the application releases the user interaction call (step 29) and the SIP session with the MRFC is released also (steps 30-33). The application releases the call to the user (step 34) and the call is released in the network.

6 Conclusion

OSA is seen by many mobile network equipment manufacturers as the successor of intelligent networks and CAMEL. That is why most service platform product lines now include an OSA gateway. By OSA interface deployment, operators can increase traffic on their networks, and hence revenue, by offering new services. Service providers can increase the number of users by offering attractive value-added services using connectivity with telecom networks. Users receive benefits through transparency and access to media services.

The paper investigates the interoperability of third party controlled user interactions and media services in managed all IP-based multimedia networks. The third party applications access user interaction functions in the network using

Table 1. Bisimulation Relation between OSA User Interaction and SIP Session

Transition in T_{AppUI}	Transition in T_{SIP}
Null *createUI* Active	*Idle* $INVITE$ *wait*200_{INVITE}, *wait*200_{INVITE} 200_{INVITE} *Established*
Null *reportEventNotification* Active	*Idle* $INVITE$ $Wait200_{INVITE}$, $Wait200_{INVITE}$ 200_{INVITE} *Established*
Active *sendInfoReq[!f]* Active	*Established* $INFO$ $Wait200_{INFO}$, $Wait200_{INFO}$ 200_{INFO} *Established*
Active *sendInfoRes[!f]* Active	*Established* $INFO$ $Wait200_{INFO}$, $Wait200_{INFO}$ 200_{INFO} *Established*
Active *sendInfoAndCollectReq[!f]* Active	*Established* $INFO$ $Wait200_{INFO}$, $Wait200_{INFO}$ 200_{INFO} *Established*
Active *sendInfoAndCollectRes[!f]* Active	*Established* $INFO$ $Wait200_{INFO}$, $Wait200_{INFO}$ 200_{INFO} *Established*
Active *sendInfoErr[!f]* Active	*Established* $INFO$ $Wait200_{INFO}$, $Wait200_{INFO}$ 200_{INFO} *Established*
Active *sendInfoAndCollectErr[!f]* Active	*Established* $INFO$ $Wait200_{INFO}$, $Wait200_{INFO}$ 200_{INFO} *Established*
Active *release* Null	*Established* BYE $Wait200_{BYE}$, $Wait200_{BYE}$ 200_{BYE} *Idle*
Active *sendInfoReq[f]* ReleasePending	*Established* $INFO$ $Wait200_{INFO}$, $Wait200_{INFO}$ 200_{INFO} *Established*
Active *sendInfoRes[!f]* ReleasePending	*Established* $INFO$ $Wait200_{INFO}$, $Wait200_{INFO}$ 200_{INFO} *Established*
ReleasePending *sendInfoErr[f]* Active	*Established* $INFO$ $Wait200_{INFO}$, $Wait200_{INFO}$ 200_{INFO} *Established*
ReleasePending *sendInfoErr[!f]* ReleasePending	*Established* $INFO$ $Wait200_{INFO}$, $Wait200_{INFO}$ 200_{INFO} *Established*
ReleasePending *sendInfoRes[f]* Finished	*Established* $INFO$ $Wait200_{INFO}$, $Wait200_{INFO}$ 200_{INFO} *Established*
ReleasePending *userInteractionAborted* Finished	*Established* $INFO$ $Wait200_{INFO}$, $Wait200_{INFO}$ 200_{INFO} *Established*
ReleasePending *release* Null	*Established* BYE $Wait200_{BYE}$, $Wait200_{BYE}$ 200_{BYE} *Idle*
Finished *release* Null	*Established* BYE $Wait200_{BYE}$, $Wait200_{BYE}$ 200_{BYE} *Idle*

Table 1. Continued

Transition in T_{AppUI}	Transition in T_{SIP}
Active *sendInfoAndCollectReq[f]* ReleasePending	$Established\ INFO\ Wait200_{INFO}$, $Wait200_{INFO}\ 200_{INFO}\ Established$
ReleasePending *sendInfoAndCollectErr[f]* Active	$Established\ INFO\ Wait200_{INFO}$, Wait$200_{INFO}$ 200_{INFO} Established
ReleasePending *sendInfoAndCollectRes[!f]* ReleasePending	$Established\ INFO\ Wait200_{INFO}$, Wait$200_{INFO}$ 200_{INFO} Established
ReleasePending *sendInfoAndCollectErr[!f]* ReleasePending	$Established\ INFO\ Wait200_{INFO}$, $Wait200_{INFO}\ 200_{INFO}\ Established$
ReleasePending *sendInfoAndCollectRes[f]* Finished	$Established\ INFO\ Wait200_{INFO}$, $Wait200_{INFO}\ 200_{INFO}\ Established$
ReleasePending *userInteractionAborted* Finished	$Established\ INFO\ Wait200_{INFO}$, $Wait200_{INFO}\ 200_{INFO}\ Established$
ReleasePending *release* Null	$Established\ BYE\ Wait200_{BYE}$, $Wait200_{BYE}\ 200_{BYE}\ Idle$
Finished *release* Null	$Established\ BYE\ Wait200_{BYE}$, $Wait200_{BYE}\ 200_{BYE}\ Idle$
Active *sendInfoReq[f]* Finished	$Established\ INFO\ Wait200_{INFO}$, $Wait200_{INFO}\ 200_{INFO}\ Established$
Finished *release* Null	$Established\ BYE\ Wait200_{BYE}$, $Wait200_{BYE}\ 200_{BYE}\ Idle$
Active *userInteractionAborted* Finished	$Established\ INFO\ Wait200_{INFO}$, $Wait200_{INFO}\ 200_{INFO}\ Established$
Finished *release* Null	$Established\ BYE\ Wait200_{BYE}$, $Wait200_{BYE}\ 200_{BYE}\ Idle$

APIs exposed by OSA gateway. The OSA gateway is responsible for transformation of API into network control protocol. We define a mapping between OSA User Interaction interface methods and SIP messages. Using a formal approach we prove behavioral equivalence between the third party application view on user interactions and the media resource control functions.

The approach is useful in testing the conformance of a black-box implementation of OSA gateway with respect to a specification, in the context of reactive systems.

Acknowledgments. The research is in the frame of Project DO-02-135/2008, funded by National Science Fund, Bulgarian Ministry of Education and Science.

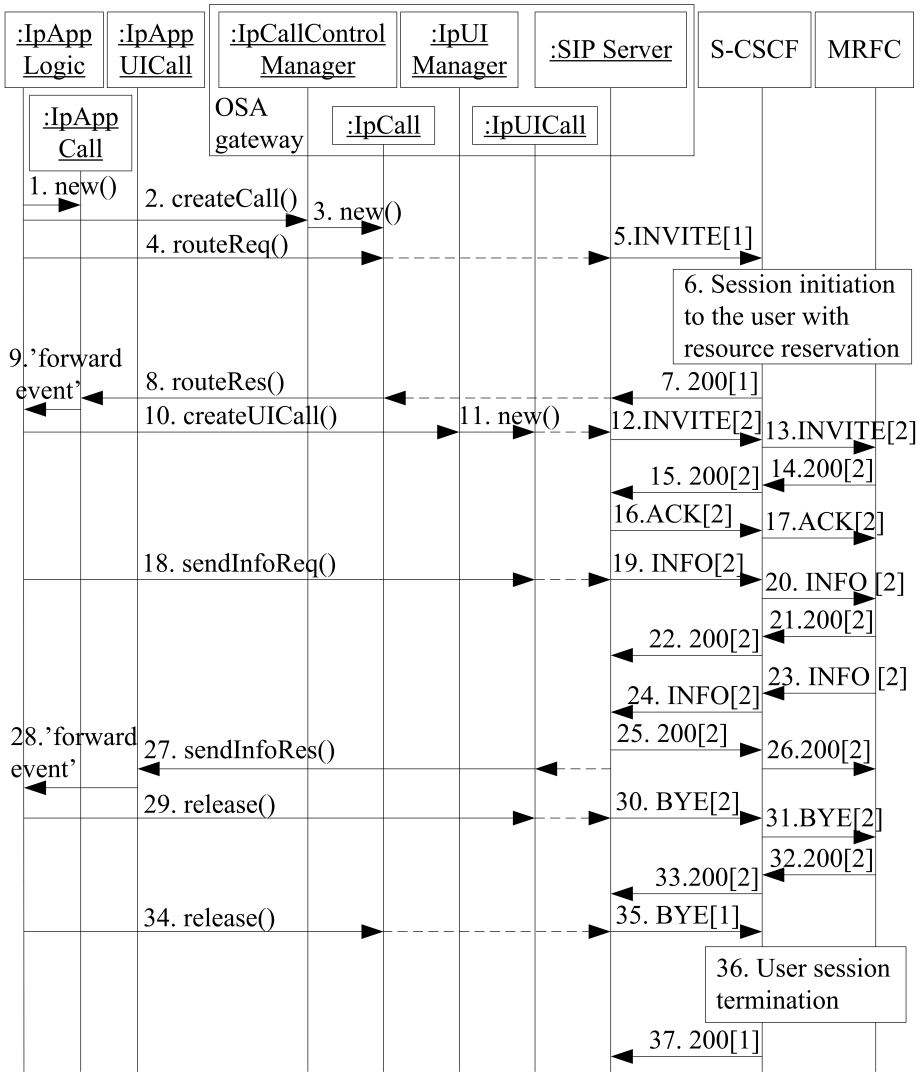

Fig. 12. Open access to media services in IMS

References

1. Poikselka, M., Mayer, G., Khartabil, H., Niemi, A.: The IMS Multimedia Concepts and Services, Third Edition. Nokia, Finland, Wiley (2009)
2. Gronbak, I.: NGN, IMS and Service Control - collected information, RI Research Note, Telenor (2006).
3. Crespi, N., Chadli, Y.: A Novel Mechanism For Media Resource Control in SIP Mobile Networks, In World Wireless Congress 2004, San Francisco, USA, http://www.it-sudparis.eu/dpt/rs2m/ncpub/2004/wwc, (2004)

4. Burke, D., O'Flanagan, D.: An IMS Application Example Based on SIP Servlets and VoiceXML, http://dev2dev.bea.com/pub/a/2006/06/ims-sip-voicexml.html, (2006)
5. Burger, E., Van Dike, J., Spitzer, A.: Basic Network Media Services with SIP, RFC 4240, (2005)
6. Burke, D., Scott, M., SIP Interface to VoiceXML Media Services, RFC 5552 (2009)
7. 3GPP TE 23.278 Customized Applications for Mobile network Enhanced Logic (CAMEL) Phase 4; stage 2; IM CN Interworking, Release 9, v9.0.0 (2009)
8. 3GPP TS 29.198-5 Open Service Access (OSA); Application Programming Interface (API); Part 5: User Interaction Service Capability Feature (SCF), v9.0.0 (2009)
9. 3GPP TR 29.998-05-1 Open Service Access; Application Programming Interface (API) Mapping for OSA: Part 5: User Interaction Service Mapping; Subpart 1: API to CAP Mapping, Release 9, v9.0.0 (2009)
10. 3GPP TR 29.998-05-5 Open Service Access; Application Programming Interface (API) Mapping for OSA: Part 5: User Interaction Service Mapping; Subpart 4: API to SMS Mapping, Release 9, v9.0.0 (2009)
11. 3GPP TS 24.229 IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3; Release 9, v9.0.0 (2009)
12. 3GPP TS 23.218 IP Multimedia (IM) session handling; IM call model; Stage 3; Release 9, v9.0.0 (2009)
13. Chena, X., Nicola, R.: Algebraic characterizations of trace and decorated trace equivalences over tree-like structures, In: Theoretical Computer Science, pp. 337-361 (2001)
14. Panangaden, P.: Notes on Labelled Transition Systems and Bisimulation, Retrieved from http://www.cs.mcgill.ca/ prakash/Courses/comp330/Notes/lts09.pdf (2004)