BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **10**, No 1

Sofia • 2010

Open Access to Call and Session Control in Mobile Networks

Evelina Pencheva, Ivaylo Atanasov

Technical University of Sofia, 1794 Sofia
E-mails: enp@tu-sofia.bg iia@tu-sofia.bg

Abstract: *The paper presents a study on implementation aspects of a server, providing application programming interfaces to functions for call control and data session control in mobile networks. The value-added service logic in mobile networks is provided on Customized Application for Mobile services Enhanced Logic (CAMEL) platform. We suggest an approach to modeling the server behavior that supports Open Service Access (OSA) interfaces of and "talks" CAMEL Application Part (CAP) protocol toward the network. The approach is presented in a formal way as labeled transition system which corresponds to the model representing the application view on the behavior of call and data session objects. It is proved by bisimulation relation between the labeled transition systems that the suggested call state model and data session state model for the control protocol expose equivalent behavior compared with those of the standardized call and data session state models as seen by OSA applications.*

Keywords: *Open service access, call control, data session control, labeled transition systems, behavioral equivalence.*

I. Introduction

Open Service Access (OSA) is an open interface that allows third party applications to invoke communication functions in a public network. Network functions accessed by applications are called Service Capability Features (SCF) [1]. Application programming interfaces are defined for each OSA SCF exposed toward the applications. The OSA Generic Call Control SCF provides functions for simple call control which allows to setup traditional two-party calls [2]. The OSA Data

Session Control SCF allows setting up, releasing, and managing packet data sessions [3].

The SCF interface toward 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 toward 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 toward applications and control protocols toward the network is called Service Capability Server (SCS). It is considered as a special type of application server. The OSA SCS deployed in mobile networks must support CAP protocol and has to realize session state models that correspond both to the application view and control protocol. Fig.1 shows the deployment of OSA SCS in CAMEL network.

For each call a Basic Call State Model (BCSM) at the Mobile services Switching Center (MSC) is triggered [4]. The OSA SCS needs to maintain a call state model that corresponds to the OSA application view of the call object. While the BCSM and model, representing the application view on call states, are standardized, there are no recommendations for the call state model in the OSA SCS which is regarded to be an implementation issue.

The packet session state is monitored at the Serving GPRS Support Node (SGSN). The OSA SCS has to maintain a session state model that corresponds to both the packet session state at the SGSN and application view on data session object. Instead of standardizing the behavior of OSA SCS, the Third Generation Partnership Project (3GPP) provides only mapping of OSA Data Session Control interfaces onto CAP protocol [5]. A mapping of OSA Generic Call Control interfaces onto CAP protocol is also available [6].

Some publications [7, 8], concerning OSA SCS implementation, focus on aspects related to the application programming interfaces, while other authors [9, 10] discuss evaluation of conformance of the basic call and session control mechanisms in the network out of the application context.

Our research aims to suggest an approach to modeling call and data session states in an OSA SCS. The call and data session state models have to expose equivalent behavior to that of the corresponding OSA interface objects as seen from application point of view. To prove the behavior equivalence we present the suggested state models in a formal way as Labeled Transition Systems (LTS) and use the concepts of *bisimulation* and *homomorphism* [11].

The rest of the paper is organized as follow. Section II briefly introduces the necessary notational background for labeled transition systems and behavioral equivalences. Section III presents an approach to modeling the originating and terminating call states that conforms to the OSA application view of a two party call. A model that represents the call states in case of application initiated call is described. Section IV presents an approach to modeling the packet session states that is synchronized with the application view on the data session object. Having in mind the implicit behavioral recommendations which are based on the available

mapping of OSA interface class methods onto CAP messages, we provide adequate conditions on transformations in order to preserve the logic equivalence. The paper is concluded by sketching out an approach to modeling the session states in IP-based multimedia networks.

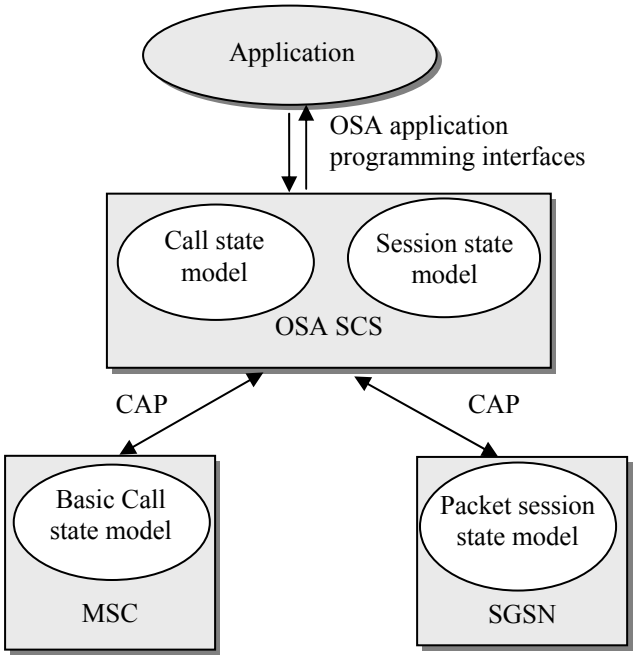

Fig.1. Deployment of OSA in circuit- and packet-switched mobile networks

II. Labeled transition systems and behavioral equivalence

Definition 1. A Label Transition System (LTS) is a quadruple $(S, \text{Act}, \rightarrow, s_0)$, where:

S is countable set of states,

Act is a countable set of elementary actions,

$\rightarrow \subseteq S \times \text{Act} \times S$ is a set of transitions, and

$s_0 \in S$ is the set of initial states.

A labeled transition system $(S, \text{Act}, \rightarrow, s_0)$ is called *finitely branching*, if for any state $s \in S$, set $\{(a, s'), a \in \text{Act} \mid (s, a, s') \in \rightarrow\}$ 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 $\mu = a_1, a_2, ..., a_n : \exists\ s_1, s_2, \ldots, s_n$, such that $s \xrightarrow{a_1} s_1 ... \xrightarrow{a_n} s_n$;

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

$\overset{\hat{\mu}}{\Longrightarrow}$ means $\Rightarrow$ 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 $h(\rightarrow) = \{h(s) \overset{a}{\rightarrow}' h(s') \mid s \overset{a}{\rightarrow} s'\}$;

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:

- $s \overset{a}{\rightarrow} s'$ implies $\exists$ t': $t \overset{a}{\rightarrow}' t'$ and $s'\ U\ t'$;
- $t \overset{a}{\rightarrow}' t'$ implies $\exists\ s'$: $s \overset{a}{\rightarrow} s'$ and $s'\ U\ t'$.

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

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 *abstraction homomorphism* if h is a surjective transition system homomorphism satisfying the following:

$\forall\ s_1 \in S,\ a \in$ Act and $s_2' \in S'$, $h(s_1) \overset{a}{\rightarrow}' s_2'$ implies $\exists\ s_2 \in S$: $s_1 \overset{a}{\rightarrow} s_2$ and $h(s_2) = s_2'$.

The following statement is true for two labeled transition systems:

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:

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

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

III. Call state models

The Application Server (AS) that hosts OSA applications can play different roles. For example, acting as a proxy server provisioning call baring services, the AS receives incoming request and proxies it back to the MSC which then forwards it toward the destination. Another case is in the context of an alarm call when the AS generates a request and sends it to the MSC which then forwards it toward destination.

First, we will consider the case when the application provides control on network initiated call.

A. Application control on terminating and originating parties

If the application logic is applied for the originating party, the OSA SCS needs to maintain a call state model that corresponds to the CAMEL Originating-Basic Call State Model (O-BCSM). If the application logic is applied for the terminating party, the OSA SCS needs to maintain a call state model that corresponds to the CAMEL Terminating-Basic Call State Model (T-BCSM).

A model representing the OSA application view on a call object is defined in [2]. According to this model, the call can be in one of the following states: Null (no call), NoParty (call object exists with no party connected), 1PartyInCall (one party in call), 2PartiesInCall (established call with two parties), NetworkReleased (call is terminated in the network and the application can only gather call information), Finished (call is terminated in the network and no call information can be gathered), ApplicationReleased (the call is terminated by the application). From call control point of view, there is no difference between the Null and NoParty states, as they reflect specifics of object-oriented technology. In the NetworkReleased, Finished and ApplicationReleased states, there are no parties connected to the call, hence from the call control point of view, these states can be labeled with the same label Null_{GCC}.

Let us denote by $T_{GCC} = (S_{GCC}, \text{Act}_{GCC}, \rightarrow_{GCC}, s_0)$ a labeled transition system representing the OSA application view on call object states where:

S_{GCC} = {$Null_{GCC}$, 1PartyInCall, 2PartiesInCall };
Act_{GCC}= {callEventNotify, connectionToCalledPartyUnsuccessful, answer,
release, callEnded};
$\rightarrow_{GCC}$ = { $Null_{GCC}$ callEventNotify 1PartyInCall,
1PartyInCall connectionToCalledPartyUnsuccessful $Null_{GCC}$,
1PartyInCall answer 2PartiesInCall,
2PartiesInCall release $Null_{GCC}$,
2PartiesInCall callEnded $Null_{GCC}$,
1PartyInCall callEnded $Null_{GCC}$};
s_0 = {$Null_{GCC}$}.

The T-BCSM consists of four states in which service logic can interfere the processing of the terminating party. In T-Null state, the terminating party is idle, and the authority to route the call to the terminating party is verified, (e.g. business group restrictions, restricted incoming calls, or bearer capability compatibility). In TermCallHandling state, terminating resource is informed of incoming call, an indication is sent to the O-BCSM that the terminating party is alerted, and the call is waited to be answered. In T-Active state, the connection is established between originating and terminating party. Default handling of the exception condition is being provided in T-Exception state.

Let us denote by T_{TBCSM} = (S_{TBCSM}, Act_{TBCSM}, $\rightarrow_{TBCSM}$, s_0') a labeled transition system, representing the CAMEL T-BCSM where:

S_{TBCSM} = {T-Null, TerminatingCallHandling, T-Active, T-Exception};
Act_{TBCSM}= {TerminatingAttemptAuthorized, Tanswer, Tdisconnect, Tabandon,
Tbusy, TnoAnswer, releaseResources};
$\rightarrow_{TBCSM}$ = { T-Null TerminatingAttemptAuthorized TerminatingCallHandling,
TerminatingCallHandling Tanswer T-Active,
TerminatingCallHandling Tabandon T-Null,
T-Active Tdisconnect T-Null,
TerminatingCallHandling Tbusy T-Exception,
TerminatingCallHandling TnoAnswer T-Exception,
T-Exception releaseResources T-Null};
s_0' = {T-Null}.

The O-BCSM consists of four states in which service logic can interfere the processing of the originating party. In O-Null state, the originating party is idle, the authority/ability to place the call with given properties (e.g. bearer capability, subscription restrictions) is verified, and initial information package/dialing string (e.g. service codes, prefixes, dialed address digits) is collected from originating party. In O-Alerting state, information is analyzed and/or translated according to the dialing plan to determine routing address and call type, routing address and call type are interpreted, the next route is selected, and the originating party waits for the terminating party to answer. In O-Active state, the connection is established between originating and terminating parties. Default handling of the exception condition is being provided in the O-Exception state.

Let us denote by T_{OBCSM} = (S_{OBCSM}, Act_{OBCSM}, $\rightarrow_{OBCSM}$, s_0'') a labeled transition system, representing the CAMEL O-BCSM where:

S_{OBCSM} = {O-Null, AnalysisRoutingAlerting, O-Active, O-Exception;
Act_{OBCSM}= {CollectedInfo, Oanswer, Odisconnect, Oabandon, Obusy, OnoAnswer, releaseResources};
$\rightarrow_{OBCSM}$ = { O-Null CollectedInfo AnalysisRoutingAlerting,
AnalysisRoutingAlerting Oanswer O-Active,
AnalysisRoutingAlerting Oabandon O-Null,
O-Active O_disconnect O-Null,
AnalysisRoutingAlerting Obusy O-Exception,
AnalysisRoutingAlerting OnoAnswer O-Exception,
O-Exception releaseResources O-Null};
s_0'' = {O-Null}.

Proposition 1. The labeled transition systems T_{GCC}, T_{TBCSM} and T_{TBCSM} are bisimular.

P r o o f. To prove that the systems expose equivalent observable behavior, it has to be found a common image under abstraction homomorphism. The homomorphism between the states of T_{GCC}, T_{TBCSM} and T_{TBCSM} is shown in Table 1, and the transition homomorphism is shown in Table 2. The state homomorphism is based on the common abstraction they represent, while the transition homomorphism is based on the mapping of OSA Generic Call Control interface methods onto CAP messages.

Table 1 State homomorphism for T_{GCC}, T_{TBCSM} and T_{TBCSM}

S_{GCC}	S_{TBCSM}	S_{OBCSM}	Common abstraction
$Null_{GCC}$,	T-Null	O-Null	Represents a condition where call processing is not active
1PartyInCall	TerminatingCallHandling	AnalysisRoutingAlerting	Represents a two party call in the setup phase
1PartyInCall	T-Exception	O-Exception	Represents an exception situation in the call setup phase
2PartiesInCall	T-Active	O-Active	Represents a stable two party call

Having the presented homomorphism, it can be formally deduced that T_{GCC}, T_{TBCSM} and T_{TBCSM} are bisimilar, i.e., they expose equivalent behavior.

It has to be noted, that in the formal definitions of T_{GCC}, T_{TBCSM} and T_{TBCSM} systems, the transitions related to route failure and connection failure are not included for simplicity. The refinement of the model with those transitions does not change the proof of behavioral equivalence because a common abstraction for those transitions also can be identified following the mapping in [5].

B. Application-initiated call

In case of application initiated call, the application connects both parties to the call. In this case, when the application creates the call object, it can request routing of the call to the remote party indicated by the target address. The call object moves to the

RoutingToDestination state where the application waits for answer from the remote party.

Table 2. Transition homomorphism for T_{GCC}, T_{TBCSM} and T_{TBCSM}

$\rightarrow_{GCC}$	$\rightarrow_{TBCSM}$	$\rightarrow_{OBCSM}$	Common abstraction
$Null_{GCC}$ callEventNotify 1PartyInCall	T-Null TerminatingAttempt-Authorized TerminatingCallHandling	O-Null CollectedInfo AnalysisRoutingAlerting	Terminating call indication
1PartyInCall answer 2PartiesInCall	TerminatingCallHandling Tanswer T-Active	AnalysisRoutingAlerting Oanswer O-Active	The called party answers.
1PartyInCall callEnded $Null_{GCC}$	TerminatingCallHandling Tabandon T-Null	AnalysisRoutingAlerting Oabandon O-Null	Indication that the calling party abandons the call.
2PartiesInCall callEnded $Null_{GCC,}$	T-Active Tdisconnect T-Null	O-Active Odisconnect O-Null	Indication for disconnect in the network.
2PartiesInCall release $Null_{GCC}$	T-Active Tdisconnect T-Null	O-Active Odisconnect O-Null	Indication for application-initiated disconnect.
1PartyInCall connectionTo-CalledParty-Unsuccessful $Null_{GCC}$	TerminatingCallHandling Tbusy T-Exception	AnalysisRoutingAlerting Obusy O-Exception	Indication that the called party is busy.
1PartyInCall connectionToCalled-PartyUnsuccessful $Null_{GCC}$	TerminatingCallHandling TnoAnswer T-Exception	AnalysisRoutingAlerting OnoAnswer O-Exception	Indication that the called party does not answer.
1PartyInCall callEnded $Null_{GCC}$	T-Exception releaseResources T-Null	O-Exception releaseResources O-Null	Indication for the end of the call.

Let us denote with $T_{GCC}' = (S_{GCC}', Act_{GCC}', \rightarrow_{GCC}', s_0')$ a labeled transition system representing the OSA application view on a call with two parties initiated by the application where:

$S_{GCC}' = \{Null_{GCC}, RoutingToDestination, 1PartyInCall, 2PartiesInCall\}$;

$Act_{GCC}' = \{$ routeReq, answer, connectionToCalledPartyUnsuccessful, release, callEnded$\}$;

$\rightarrow_{GCC}' = \{Null_{GCC}$ routeReq RoutingToDestination,
RoutingToDestination answer 1PartyInCall,
1PartyInCall routeReq 1PartyInCall,
1PartyInCall answer 2PartiesInCall,
2PartiesInCall callEnded $Null_{GCC,}$,
2PartiesInCall release $Null_{GCC}$,
RoutingToDestination connectionToCalledPartyUnsuccessful $Null_{GCC,}$
1PartyInCall connectionToCalledPartyUnsuccessful $Null_{GCC}$,

1PartyInCall callEnded Null_{GCC}};

$s_0' = \{\text{Null}_{GCC}\}$.

In case of application-initiated call, the network considers both call parties as terminating. The call model in OSA SCS suggested by us is based on triggering of two T-BCSMs for each of the call parties.

Let us denote with $T = (S_{APPCALL}, \text{Act}_{APPCALL}, \rightarrow_{APPCALL}, s_0')$ a labeled transition system representing the network model of an application-initiated call where:

$S_{APPCALL}$ = {Null, TerminatingCallHandling′, 1Active, TerminatingCallHandling″, 2Active, Exception};

$\text{Act}_{APPCALL}$= {$\text{TerminatingAttemptAuthorized}_1$, answer_1, disconnect, abandon, busy_1, noAnswer_1, $\text{TerminatingAttemptAuthorized}_1$, answer_2, busy_2, noAnswer_2, releaseResources};

$\rightarrow_{APPCALL}$ = {Null $\text{TerminatingAttemptAuthorized}_1$ TerminatingCallHandling′,
TerminatingCallHandling′ busy_1 Exception,
TerminatingCallHandling′ noAnswer_1 Exception,
TerminatingCallHandling′ answer_1 1Active,
1Active $\text{TerminatingAttemptAuthorized}_2$ TerminatingCallHandling″,
TerminatingCallHandling″ answer_2 2Active,
2active Disconnect Null,
2active release Null,
TerminatingCallHandling″ abandon Null,
TerminatingCallHandling″ busy_2 Exception,
TerminatingCallHandling″ noAnswer_2 Exception,
Exception releaseResources Null}

$s_0' = \{\text{Null}\}$.

Proposition 2. The labeled transition systems T_{GCC}' and $T_{APPCALL}$ are bisimilar.

P r o o f. The common homomorphism abstraction for T_{GCC}' and $T_{APPCALL}$ is shown in Table 3 and Table 4 where the state and transition homomorphisms are presented correspondingly. The identified abstractions allow to deduce that T_{GCC}' and $T_{APPCALL}$ are bisimilar.

Table 3 State homomorphism for T_{GCC}' and $T_{APPCALL}$

S_{GCC}	$S_{APPCALL}$	Common abstraction
Null_{GCC}	Null	No parties are connected to the call
RoutingToDestination	TerminatingCallHandling′	Connection establishment to first party in a call
1PartyInCall	1Active	A call with one party connected
1PartyInCall	TerminatingCallHandling″	A call with connected one party in a process of connection establishment to the second party
2PartiesInCall	2Active	A call with two parties connected
1PartyInCall	Exception	An exception situation in connection establishment to one party in a call

IV. Data session state model

While the OSA Generic Call Control SCF provides open access to call control functions in circuit switched networks, the Data Session Control SCF provides access to functions for data session control in packet switched networks. In [3], a model representing the application view on data session states is defined. According to that model, the data session object can be in one of the following states. In Idle state, the data session object is not created. The Setup state is entered when the reportNotification method indicates that a data session, which the application is interested in, is setup. In Setup state, the application can request session establishment to the remote party by invoking connectReq method. In Established state, the application is notified that the session is established. In NetworkReleased state, the session is terminated and the application can gather session information. In ApplicationReleased state, the data session is terminated by the application. From data session control point of view, the Idle, Network released and Application released states are equivalent, as far as the data session is neither in establishing nor active phase.

Before data can be sent or received, a packet data protocol context (PDP context) for the user has to be established. The PDP context assigns an IP address for the communication and associates it with the user identities. The CAMEL PDP context state model represents the process of establishing PDP contexts for data communications. It is comparable to the O-BCSM and T-BCSM in CAMEL. Such model is triggered for each user which wants to participate in packet data session. The model consists of three states. In Idle state, there is no PDP context active (neither sending nor receiving data). A request to send or receive data causes a transition to PDP_context_setup state. When the SGSN is notified that the PDP context is set up, a transition to the PDP context_established state takes place. It is possible that while the user is sending or receiving data, he or she moves from one routing area to another. A direct transition form Idle state to PDP_context_stablished state is possible during routing area update with SGSN change. A PDP can be deactivated either by request of the user or by the network, causing transition to Idle state. An exception situation during PDP context setup or in PDP context established state also results in transition to Idle state.

We suggest a session model that reflects both the application view on data session object and the CAMEL PDP context setup model.

Let us denote with $T_{DSC} = (S_{DSC}, Act_{DSC}, \rightarrow_{DSC}, s_0)$ a labeled transition system, representing the OSA application view on data session state where:

S_{DSC} = {$Idle_{DSC}$, $Setup_{DSC}$, $Established_{DSC}$};

Act_{DSC} = {dataSessionSetup, dataSessionEstablished, connectReq, release, dataSessionDisconnected, dataSessionSupervisionEvent};

$\rightarrow_{DSC}$ = {$Idle_{DSC}$ dataSessionSetup $Setup_{DSC}$,
$Setup_{DSC}$ connectReq $Setup_{DSC}$
$Setup_{DSC}$ dataSessionEstablished $Established_{DSC}$,
$Idle_{DSC}$ dataSessionEstablished $Established_{DSC}$,
$Setup_{DSC}$ dataSessionDisconnected $Idle_{DSC}$,
$Setup_{DSC}$ release $Idle_{DSC}$,

Established_{DSC} dataSessionDisconnected Idle_{DSC},
Established_{DSC} release Idle_{DSC}
Established_{DSC} dataSessionSupervisionEvent $\text{Established}_{DSC}\}$;
$s_0 = \{\text{Idle}_{DSC}\}$.

Table 4. Transition homomorphism for T_{GCC}' and $T_{APPCALL}$

$\rightarrow_{GCC}'$	$\rightarrow_{APPCALL}$	Common abstraction
Null_{GCC} route Req RoutingToDestination	Null $\text{TerminatingAttemptAuthorized}_1$ TerminatingCallHandling′	Indication of terminating call to the first party.
RoutingToDestination answer 1PartyInCall	TerminatingCallHandling′ answer_1 1Active	The first party in call answers.
1PartyInCall routeReq 1PartyInCall	1Active $\text{TerminatingAttemptAuthorized}_1$ TerminatingCallHandling″	Indication of terminating call to the second party.
1PartyInCall answer 2PartiesInCall	TerminatingCallHandling″ answer_2 2Active	The second party in call answers.
2PartiesInCall callEnded Null_{GCC}	2active Disconnect Null	Network-released call.
2PartiesInCall release Null_{GCC}	2active release Null	Application releases the call.
1PartyInCall callEnded Null_{GCC}	TerminatingCallHandling″ abandon Null	First party abandons the call
RoutingToDestination connectionToCalledPartyUnsuccessful Null_{GCC}	TerminatingCallHandling′ busy_1 Exception	Indication that the first party is busy
RoutingToDestination connectionToCalledPartyUnsuccessful Null_{GCC}	TerminatingCallHandling′ noAnswer_1 Exception	Indication that the first party does not answer
1PartyInCall connectionTo-CalledPartyUnsuccessful Null_{GCC}	TerminatingCallHandling″ busy_2 Exception	Indication that the second party is busy
1PartyInCall connectionTo-CalledPartyUnsuccessful Null_{GCC}	TerminatingCallHandling″ noAnswer_2 Exception	Indication that the second party does not answer
1PartyInCall callEnded Null_{GCC}	Exception releaseResources Null	Indication of the call end

The proposed session model includes two PDP context setup models for each party in the session. To distinguish between the two types of routing area change we use indexing. With $\text{ChangeOfPosition}_{O1}$ we denote the transition when the originating party changes the routing area and SGSN. With $\text{ChangeOfPosition}_{O2}$ we denote the transition when the originating party performs routing area update without SGSN change. The indexes for the mobility of the terminating party are similar.

Let us denote by $T_{PDP} = (S_{PDP}, \text{Act}_{PDP}, \rightarrow_{PDP}, s_0')$ a labeled transition representing the session state for both parties involved where:

S_{PDP} = {$Idle_{PDP}$, $PDPcontextSetup_O$, $PDPcontextEstablished_O$,
$PDPcontextSetup_T$, $PDPcontextEstablished_T$ };

Act_{PDP} = {$PDPctxtEstablishment_O$, $PDPctxtAck_O$, $disconnect_O$, $exception_O$,
$ChangeOfPositionCtxt_{O1}$,$ChangeOfPositionCtxt_{O2}$,
$PDPctxtEstablishment_T$, $PDPctxtAck_T$, $disconnect_T$, $exception_T$,
$ChangeOfPositionCtxt_{T1}$, $ChangeOfPositionCtxt_{T2}$};

$\rightarrow_{PDP}$ = {$Idle_{PDP}$ $PDPctxtEstablishmen_O$ $PDPcontextSetup_O$,
$PDPcontextSetup_O$ $PDPctxtAck_O$ $PDPcontextEstablished_O$,
$PDPcontextSetup_O$ $exception_O$ $Idle_{PDP}$,
$Idle_{PDP}$ $ChangeOfPositionCtxt_{O1}$ $PDPcontextEstablished_O$,
$PDPcontextEstablished_O$ $ChangeOfPositionCtxt_{O2}$
$PDPcontextEstablished_O$,
$PDPcontextEstablished_O$ $exception_O$ $Idle_{PDP}$,
$PDPcontextEstablished_O$ disconnect $Idle_{PDP}$,
$PDPcontextEstablished_O$ $PDPctxtEstablishment_T$ $PDPcontextSetup_T$,
$PDPcontextSetup_T$ $exception_T$ $Idle_{PDP}$,
$PDPcontextSetup_T$ $ChangeOfPositionCtxt_{O1}$ $PDPcontextSetup_T$,
$PDPcontextSetup_T$ $ChangeOfPositionCtxt_{O2}$ $PDPcontextSetup_T$,
$PDPcontextSetup_T$ $PDPctxtAck_T$ $PDPcontextEstablished_T$,
$PDPcontextEstablished_T$ $ChangeOfPositionCtxt_{T1}$
$PDPcontextEstablished_T$
$PDPcontextEstablished_T$ $ChangeOfPositionCtxt_{T2}$
$PDPcontextEstablished_T$
$PDPcontextEstablished_T$ $ChangeOfPositionCtxt_{O1}$
$PDPcontextEstablished_T$
$PDPcontextEstablished_T$ $ChangeOfPositionCtxt_{O2}$
$PDPcontextEstablished_T$
$PDPcontextEstablished_T$ $exception_T$ $Idle_{PDP}$,
$PDPcontextEstablished_T$ $exception_O$ $Idle_{PDP}$,
$PDPcontextEstablished_T$ disconnect $Idle_{PDP}$}

$s_0' = \{Idle_{PDP}\}$.

Proposition 3. The labeled transition systems T_{DSC}, and T_{PDP} are weakly bisimilar.

P r o o f. Let us build the relation $U = S_{DSC} \times S_{PDP}$ as

$$U = \{(Idle_{DSC}, Idle_{PDP}), (Setup_{DSC}, PDPcontextSetup_O), (Established_{DSC}, PDPcontextEstablished_T)\}$$

Given the mapping of Data Session Control interface methods onto CAP messages in [5], we build the following homomorphism relation h between Act_{DSC} and Act_{PDP}, presented in Table 5.

Based on the common abstraction the related transitions can be labeled by the same labels. The proof that U is a weak bisimulation relation between the states of T_{DSC} and T_{PDP} follows from $h(Act_{DSC}) = Act_{PDP}$ and the corresponding transition relations shown in Table 6.

Table 5. Homomorphism between Act_{DSC} and Act_{PDP}

Act_{DSC}	Act_{PDP}	Common abstraction
dataSessionSetup	$PDPctxtEstablishment_O$ $PDPctxtAck_O$ $ChangeOfPositionCtxt_{O1}$	Indication of PDP context establishment for the originating party
connectReq	$PDPctxtEstablishment_T$	Request for PDP context establishment for the terminating party
dataSessionEstablished	$PDPctxtAck_T$	Indication of established PDP contexts for both parties
Release	disconnect	Application-initiated session termination and PDP deactivation
dataSessionDisconnected	disconnect	Network-initiated session termination and PDP deactivation
dataSessionSupervisionEvent	$ChangeOfPositionCtxt_{T1}$ $ChangeOfPositionCtxt_{O2}$ $ChangeOfPositionCtxt_{T2}$	Indication of session related event

Table 6 Bisimulation relation between the states of T_{DSC} and T_{PDP}

$s_1 \overset{a}{\Longrightarrow} s_2 \mid s_1, s_2 \in S_{DSC}$	$t_1 \overset{\hat{a}}{\Longrightarrow}' t_2 \mid t_1, t_2 \in S_{PDP}$	Abstraction
$Idle_{DSC}$ dataSessionSetup $Setup_{DSC}$	$Idle_{PDP}$ $PDPctxtEstablishmen_O$ $PDPcontextSetup_O$,	Indication of request for PDP context establishment for the originating party
$Setup_{DSC}$ dataSessionEstablished $Established_{DSC}$	$PDPcontextSetup_O$ $PDPctxtAck_O$ $PDPcontextEstablished_O$, $PDPcontextEstablished_O$ $ChangeOfPositionCtxt_{O2}$ $PDPcontextEstablished_O$ $PDPcontextEstablished_O$ $PDPctxtEstablishment_T$ $PDPcontextSetup_T$, $PDPcontextSetup_T$ $ChangeOfPositionCtxt_{O1}$ $PDPcontextSetup_T$ $PDPcontextSetup_T$ $ChangeOfPositionCtxt_{O2}$ $PDPcontextSetup_T$ $PDPcontextSetup_T$ $PDPctxtAck_T$ $PDPcontextEstablished_T$	PDP context establishment for the originating and terminating parties. During PDP context setup for the terminating party, the originating party can perform both types of routing area update
$Setup_{DSC}$ dataSessionDisconnected $Idle_{DSC}$	$PDPcontextSetup_O$ $exception_O$ $Idle_{PDP}$,	Exception situation at the originating party during PDP context establishing for the originating party
$Setup_{DSC}$ dataSessionDisconnected $Idle_{DSC}$	$PDPcontextSetup_T$ $exception_O$ $Idle_{PDP}$,	Exception situation at the originating party during PDP context establishing for the terminating party
$Setup_{DSC}$ dataSessionDisconnected $Idle_{DSC}$	$PDPcontextSetup_T$ disconnect $Idle_{PDP}$,	The originating party disconnects during PDP context setup for the terminating party

Table 6 (c o n t i n u e d)

$Setup_{DSC}$ dataSessionDisconnected $Idle_{DSC}$	PDPcontextSetup$_T$ exception$_T$ $Idle_{PDP}$	Exception situation at the terminating party during PDP context establishing for the terminating party
$Setup_{DSC}$ dataSessionDisconnected $Idle_{DSC}$	PDPcontextEstablished$_T$ exception$_O$ $Idle_{PDP}$,	Exception situation at the originating party during an established session
$Setup_{DSC}$ dataSessionDisconnected $Idle_{DSC}$	PDPcontextEstablished$_T$ exception$_T$ $Idle_{PDP}$,	Exception situation at the terminating party during an established session
$Idle_{DSC}$ dataSessionEstablished $Established_{DSC}$	$Idle_{PDP}$ ChangeOfPositionCtxt$_{O1}$ PDPcontextEstablished$_{O,}$ PDPcontextEstablished$_O$ PDPctxtEstablishment$_T$ PDPcontextSetup$_T$, PDPcontextSetup$_T$ PDPctxtAck$_T$ PDPcontextEstablished$_T$	During PDP context establishment for the terminating party, the originating party performs routing area update with SGSN change
$Established_{DSC}$ dataSessionSupervisionEvent $Established_{DSC}$	PDPcontextEstablished$_T$ ChangeOfPositionCtxt$_{O2}$ PDPcontextEstablished$_T$ PDPcontextEstablished$_T$ ChangeOfPositionCtxt$_{O1}$ PDPcontextEstablished$_T$	During established session, the originating party performs both type of routing area update
$Established_{DSC}$ dataSessionSupervisionEvent $Established_{DSC}$	PDPcontextEstablished$_T$ ChangeOfPositionCtxt$_{T1}$ PDPcontextEstablished$_T$ PDPcontextEstablished$_T$ ChangeOfPositionCtxt$_{T2}$ PDPcontextEstablished$_T$	During established session, the terminating party performs both type of routing area update
$Established_{DSC}$ dataSessionDisconnected $Idle_{DSC}$	PDPcontextEstablished$_T$ disconnect $Idle_{PDP}$	The session terminates
$Established_{DSC}$ release $Idle_{DSC}$	PDPcontextEstablished$_T$ exception$_O$ $Idle_{PDP}$ PDPcontextEstablished$_T$ exception$_T$ $Idle_{PDP}$	The application requests established session termination
$Setup_{DSC}$ release $Idle_{DSC}$	PDPcontextEstablished$_T$ exception$_O$ $Idle_{PDP}$ PDPcontextEstablished$_T$ exception$_T$ $Idle_{PDP}$	The application requests termination of establishing session

V. Conclusion

The network interface of the OSA SCS is not standardized and it is regarded as implementation detail. There is a constraint for any implementations for a single point of contact. Instead of coordinating the connections in circuit-switched and packed-switched networks, the OSA SCS controls the interface to the underlying

network. In mobile networks, bearer connections rely on CAP signaling. Interfacing between application programming interfaces requests and responses, and CAP signaling, the OSA SCS needs to care for both side state machines – one for the interface objects and another for the CAP session.

The suggested approach to modeling the call and session states in an OSA SCS supports interfaces for generic call control and data session control. While the call state model corresponds to the CAMEL BCSMs, the session state model has to reflect the procedures concerning PDP context setup required for packet sessions. More refined models can be built considering different application server roles.

Following the same approach, session state models in IP-based multimedia networks might be designed. In such networks the establishment, modification and termination of sessions with more than two parties relies on SIP. Following the mapping of OSA Multiparty Call Control SCF onto SIP signaling, models for SIP session state for originating and terminating call legs can be designed.

By presenting the models in a formal way as label transition systems, the behavior equivalence is proved, using the concept of bisimulation.

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

References

1. Hanrahan, Hu. Network Convergence, Services, Applications, Transport and Operations Support. Wiley, 2007.
2. 3GPP TS 29.198-4-2, v9.0.0, 2009, Open Service Access (OSA); Application Programming Interface (API) Part 2: Call Control Sub-part 2: Generic Call Control Service Capability Feature; (Release 9).
3. 3GPP TS 29.198-08, v9.0.0, 2009, Open Service Access (OSA); Application Programming Interface (API) Part 8: Data Session Call Control Service Capability Feature; (Release 9).
4. Noldus, R. CAMEL Intelligent Networks for GSM, GPRS and UMTS Networks. Wiley, 2006.
5. 3GPP TR 29.998-09, v9.0.0, 2009, Open Service Access (OSA); Mapping for Open Service Access; Part 8, Data Session Control Service Mapping to CAP, (Release 9).
6. 3GPP TR 29.998-04-1, v9.0.0, 2009, Open Service Access (OSA); Mapping for Open Service Access; Part 4, Call Control Service Mapping; Subpart 2: API to CAP mapping, (Release 9)
7. Imrich, Chlamtac, Hsin-Yi Lee, Yi-Bing Lin, Meng-Hsun Tsai. An OSA Service Capability Server for Mobile Services. – International Journal of Pervasive Computing and Communications, Vol. **4**, 2008, Issue 3, 268-278.
8. Stretch, R. The OSA API and Other Related Issues. – BT Technology Journal, Vol. **19**, 2001, No 1, 80-87.
9. Magedanz, T., D. Witaszek, K. Knüttel. Service Delivery Platform Options for Next Generation Networks, Approved Within The National German 3G Beyond Testbed. **http://home.intekom.com/satnac/proceedings/2004/Plenary/Magedanz.pdf**
10. Stuart, Walker. Providing IMS Services to Legacy Network Endpoints, 2009. **http://www.iec.org/newsletter/august07_1/analyst_corner.pdf**
11. Panangaden, P. Notes on Labeled Transition Systems and Bisimulation, 2009. **http://www.cs.mcgill.ca/~prakash/Courses/comp330/Notes/lts09.pdf**