MAPPING OF GENERIC OPEN INTERFACES TO NETWORK PROTOCOLS

Hristo E. Froloshki, Evelina N. Pencheva

Department of Telecommunications, Technical University – Sofia, "Kliment Ohridski" Blvd. 8 1000, Sofia, Bulgaria, phone: (+359) (2) 965 2134, e-mail: hef@tu-sofia.bg, enp@tu-sofia.bg

The paper presents a mapping of Parlay/Open Service Access (OSA) Application programming interfaces (APIs) onto underlying network protocols like ISDN user part (ISUP), CAMEL application part (CAP) and Session initiation protocol (SIP). The focus is on Call control API enabling third party applications to interfere in creation, manipulation, termination and teardown of communication sessions.

Keywords: Open service access, Call control, SIP, CAP, ISUP

1. INTRODUCTION

Next generation networks bring the promise for a multitude of attractive services, satisfying subscriber's needs. Various networking technologies are expected to cooperate and deliver valuable solutions. Industrial leaders in IT and telecommunication sectors apply the proved concept of layering to the process of application accessing network services. The concept is brought to life by 3rd generation partnership project consortium in the form of application programming interfaces (APIs), known as Parlay/OSA (Open Service Access). Each API is aimed at specific network functionality called service capability such as call-control, user interaction, mobility, data session control, network management, etc. OSA APIs provide application developers with programmability of network resources defining them in terms of objects with methods, data types and parameters on which they operate.

Present paper focuses on call-control API, and in particular its generic call control part, since it has heavy impact on all services, requiring application intervention in call creation, manipulation, termination and teardown. Third party applications are able to setup a communication sessions or play scheduled announcements to subscribers of these networks. Some of the important call control API methods are considered and brief overview of their meaning for voice-carrying networks such as circuit switched or packet switched telephony is performed. Suggestions for mapping between OSA API methods and underlying network protocol messages are presented as well. Then validation of the proposed concept is done through functional model of a software entity, capable of translating between Parlay/OSA application servers and voice networks, employing ISUP (ISDN user application part) and SIP (Session initiation part) signaling protocols. A brief overview is made for CAMEL (Customiz-

ed application for mobile services enhanced logic) enabled mobile networks, where mapping is given in [1] and telephony networks structured as intelligent one [2].

2. CALL CONTROL API AND EXPOSED NETWORK FUNCTIONALITY

Originally the Parlay/OSA APIs are intended for exclusive use in third generation mobile networks. However the 3GPP standards are open by design and allow expansion toward fixed telephony networks. In previous works call control objects, defined by the API were discussed [3], while here the stress is on implications of mapping the API's methods on network-specific protocols and the nodes, who understand them.

A primary functionality required by the call control API is a mechanism for transporting notifications about the states in call for a particular subscriber. This information allows service logic (residing in application server) to control the call (put it on hold, redirect, play a wakeup message and so on), bringing more value to the subscriber. Call related information in circuit switched networks structured as intelligent ones is accessible through INAP (Intelligent Network Application Part) protocol.

Another key functionality is the option for load-control, requested by application. The most important reason for this is to keep processing load of both the application server and the Parlay gateway within reasonable limits - otherwise there is a risk for failing the execution of services and/or degrading quality of service parameters.

The API enables applications to obtain timing information, useful for service delivery, such as: starting time of call, end time of call, time of connection to a resource, etc. It should be noted that this is possible if the underlying network (whose resources are abstracted) supports these parameters, for example mobile network supporting CAMAL application part (CAP). On contrary with circuit switched networks, SIP-based telephony networks do not provide call-related timing information. To provide this sort of information the implementation of SIP proxy has to consider the necessity of collecting call information.

Last, but not least is the feature, allowing application to request the routing of a call from originating to destination address. Here the API is influenced by options, provided by INAP and CAP messages, while SIP at first glance satisfies partially the required functionality (concerning services such as redirect or mid-call digit entry). This leads to the conclusion that some services will not be portable across different types of networks. As suggested in [4] there are certain mechanisms in SIP to transport information, related with mid-call services, but these are left to be implementation specific, which contradicts the founding idea of Parlay – to present generic network services, to application servers.

3. MAPPING API METHODS ON NETWORK PROTOCOLS PROCEDURES

The paper considers only a fraction of API's methods, since not all are appropriate for translation to underlying networks. As illustration for this statement let's consider

load control feature, requested by application. In CAP enabled mobile network this translates into call gap procedure, while in SIP network, this must be implemented in internal proxy routine. This section focuses on some methods defined for IpCallControlManager and IpCall objects and their mappings on CAP, INAP, ISUP and SIP.

3.1 CAP Mapping

The most important methods for IpCallControlManager are enable(disable)CallNotification used for declaring the application interest in states of a call. These are translated as Mobile application part messages AnyTimeModification and obtain CAMEL subscriber information. Actual call state information is transferred through callEventNotify method, which extracts it from the CAP InitialDP message. Considered IpCall methods are RouteReq – to actually setup a call the application invokes the method, which in turn is translated to CAP connect message. It carries all the necessary information for the service switching point. Here should be noted that RouteReq method assumes translation to different CAP messages (CAP Connect, Continue and ContinueWithArgument), depending on the method provided values. CAP EventReportBCSM message carries the type of event, detected in the mobile network. Its translation is invocation of RouteRes method, implemented by IpAppCall object on the application side. If the call state is disconnected, the called API method is callEnded. DeassignCall method is translated as CAP Cancel or Continue to indicate that the application is no longer interested in controlling the call. GetCallInfoReq is mapped on CAP CallInformationRequest in order to provide call related timing information.

3.2 ISUP Mapping

Direct mapping to ISUP is possible, but advantages gained from it are minor when compared with rise in requirements and complexity of network servers. This section presents some assumptions for such mapping and uses the approach used in [5] – direct mapping of the call control API on call setup signaling protocol (SIP). It was mentioned above for IpCallControlManager requirements in terms of call event notifications, which should be maintained by a call state machine, implemented in the network server itself. In contrast CAP and INAP do not pose such requirement for the network server – the basic call state model is implemented in the switch and call state information is transported by the two protocols (CAP and INAP). Functional mapping is the only way to translate the call control API in ISUP messages – this means introduction of new nodes or at least new functionality in the fixed telephony network. These ideas apply for the other considered object – IpCall – some of its methods indeed have direct translation (routeReq – IAM, release/callEnded – REL), while others (deassignCall, routeRes, getCallInformation) require availability of basic call state model implementation within the network server.

3.3 SIP Mapping

SIP has gained enough momentum in its development and deployment to be considered as a viable candidate for extension toward IN services (for the cases where these services are required in heterogeneous environments). [4] offers architecture for IN enabled SIP proxy, capable of processing INAP messages. There is also mapping between states in the two basic call state models concerning originating and terminating part and the SIP state machine, enabling provision of SIP call states through INAP. Availability of SIP call-state model makes it possible for Parlay/OSA applications to request services from a SIP proxy in a generic way. [5] offers architecture where the SIP user agent functionality is accessible directly through OSA servers. The difference between two approaches, is that in the first case intermediary protocol is used to deliver state information, from SIP server supporting call state machine, while the latter uses direct mapping, also assuming that the SIP server supports call state machine. There is no doubt that call state machine is needed in order to provide IN services, but a question arises: do we need the intermediate functions of INAP? The answers are yes – where we expect mixed environments and no – for entirely IP-based networks. Furthermore, the API is more "friendly" toward INAP than SIP.

Mapping of Call Control Manager's methods poses few difficulties when considering SIP network architecture. EnableCallNotification method requests subscription for call-related events, associated with SIP. [4] offers partial solution because events are detected and reported before the active state. SIP supports hold service, but the real problem with the active state is the transfer of information (such as dialed digits) or implementing a service as redirecting the active call on another address. The reason for this "disability" is that SIP proxy servers do not control the media stream. Methods in IpCall also present some ambiguity – RouteReq may translate into Invite message, but some parameters, concerning services such as call divert or call redirect do not have SIP equivalents, and need to be mapped functionally (which is possible if we consider the intelligent network service *call divert* and the functionality of a redirect server). Method deassignCall can be implemented internally for the INAP-enabled SIP proxy. Release is mapped on a Bye message. Methods for collecting timing information for a call also should be implemented internally.

4. PARAMETER MAPPING

This section gives the resulting parameter mappings for INAP, ISUP and SIP. For some API methods there are no direct corresponding messages and some assumptions, described in previous section are applied. Tabl.1, tabl.2, tabl.3 summarize mapping of Generic Call Control (GCC) API on to INAP, ISUP and SIP protocols correspondingly.

Tab.1. Mapping GCC API to INAP

GCC object	Method	Parameters	INAP Procedure	INAP Parameter
IpCall	routeReq()	targetAddress originatingAddress originalDestinationAddress redirectingAddress	Connect	destinationRoutingAddress callingPartyAddress originalCalledPartyID redirectingPartyID
IpApp-Call	routeRes()	eventReport	EventReportBCSM	eventTypeBCSM eventSpecificInformationBCSM
IpCall	release()	callSessionID cause	ReleaseCall	initialCallSegment Cause
IpCall	Deassign-Call()	callSessionID	-	Internal implementation for SCS
IpCall	getCallInfo-Req()	callInfoRequested	CallInforma-tionRequest	requestedInformationTypeList
IpCall Control Manager	setCallLoad Control	Duration address range	Call Gap	GapIndicators calledAddressValue

Tab.2. Mapping GCC API to ISUP

GCC object	Method	Parameters	ISUP Message	ISUP Parameter
IpCall	routeReq()	targetAddress originatingAddress originalDestinationAddress redirectingAddress	Initial Address Message	Called party number Calling party number Original called number Redirecting number
IpAppCall	routeRes()	eventReport	N/A	N/A
IpCall	release()	callSessionID cause	Release Message	Cause indicators
IpCall	deassignCall()	callSessionID	-	Internal implementation for SCS
IpCall	getCallInfo Req()	callInfoRequested	-	Internal implementation for SCS
IpCall Control Manager	setCallLoad Control	Duration address range	-	Internal implementation for SCS

Tab.3. Mapping GCC API to SIP

GCC object	Method	Parameters	SIP method	SIP Parameter
IpCall	routeReq()	targetAddress originatingAddress originalDestination-Address redirectingAddress	Invite Redirect	To: From: To: N/A
IpApp-Call	routeRes()	eventReport	EventReportBCSM (INAP-enabled SIP proxy)	eventTypeBCSM eventSpecificInformationBCSM
IpCall	release()	callSessionID cause	ReleaseCall (INAP-enabled SIP proxy)	initialCallSegment Cause
IpCall	Deassign-Call()	callSessionID	-	Internal implementation for SCS
IpCall	getCallInfo-Req()	callInfoRequested		Internal implementation for SCS
IpCCl Manager	setCallLoadControl	Duration address range		Internal implementation for SCS

5. CONCLUSIONS

The paper presents results of thorough analysis of OSA open interfaces applicability in circuit and packet switched networks. A mapping of OSA Call Control methods onto CAP, INAP, ISUP and SIP network is done. The analysis points that SIP-based telephony networks do not support the whole OSA call control functionality - it is up to implementation of SIP servers to provide it. Enhancing SIP call control features will allow external applications benefit from whole palette of network functions accessible through OSA APIs.

6. REFERENCES

[1] 3GPP TR 29.998-04-1, Application Programming Interface (API) Mapping for OSA;Part 4:Call Control Service Mapping; Subpart 1: API to CAP Mapping, V.5.0.0, 2002.

[2] 3GPP TR 29.998-04-2, Application Programming Interface (API) Mapping for OSA;
Part 4:Call Control Service Mapping; Subpart 2: API to INAP Mapping, 2002.

[3] Froloshki H., Pencheva E., *Model of OSA / Parlay Gateway For Call Control*, Proceedings of ICEST 2006,Book 1 pp. 113 - 116

[4] Gurbani V. K., Haerens F., Rastogi V., *Interworking SIP and Intelligent Network (IN) Applications* , RFC 3976,January 2005

[5] 3GPP TR 29.998-04-4, Application Programming Interface (API) Mapping for Open Service Access; Part 4: Call Control Service Mapping; Subpart 4: Multiparty Call Control ISC, 2002

[6] Xiaotao W., Schulzrinne, H., *Programmable end system services using SIP*, Proceedings of IEEE International Conference on Communications, 2003. Book 2 pp. 789- 793, 2003.