Deployment of Payment Web Service in Multimedia Networks

IVAYLO ATANASOV, EVELINA PENCHEVA, DORA MARINSKA
Faculty of Telecommunications
Technical University of Sofia
8 Kliment Ohridski blvd, 1000 Sofia
BULGARIA
iia@tu-sofia.bg, enp@tu-sofia.bg, dmarinska@gcn.bg

Abstract. Parlay X is a set of Web Services interfaces designed to provide an open access to network functions. Deployment of Web Services requires interoperability between Parlay X interfaces and Open Service Access (OSA) interfaces. The OSA gateway is responsible for the translation of OSA interfaces and the network control protocols. The paper presents a study on alternatives for deployment of Parlay X Payment Web Service in Internet Protocol based multimedia networks (IMS). It provides a transformation of Parlay X payment model to the IMS charging model based on the Diameter protocol. A formal approach to specification of OSA gateway is proposed.

Key-Words: Service Oriented Architecture, Application Programming Interfaces, Charging, Interface to protocol mapping, Labelled Transition Systems, Behavioural equivalence

1 Introduction
Opening the network using application programming interfaces (APIs) allows third party applications to access communication capabilities in public networks. A standardized architecture for open access to functions in mobile networks is Parlay/Open Service Access (OSA) [1]. The Parlay/OSA APIs expose communication functions in a neutral way for both the network technology and programming language aspects. The APIs are an abstraction from different specific protocol stacks, but the abstraction level of Parlay/OSA APIs is not oriented to what we call traditional group of web developers and this could affect usability. The Parlay X [2] is the name standing for a set of interfaces allowing access to Parlay/OSA APIs via Web Services. The applications can use Web Services’ access operations within the Parlay X gateway, where the Web Services interfaces are implemented.

There exist different deployment scenarios for Parlay X Web Services [3]. One possible scenario addresses interoperability solutions where the Parlay X functionality is an add-on to the Parlay/OSA functionality. Thus, the Parlay X gateway is connected to the OSA gateway through a Parlay/OSA interfaces, forming a kind of interface wrapping. Another scenario for Parlay X deployment addresses hybrid solutions which combine Web Service interfaces and network protocols.

In this paper, we study implementation issues of Parlay X interfaces in manageable IP-based multimedia networks (IMS). The focus is on the Parlay X interfaces supporting payment control function and their mapping onto the Diameter protocol. Based on the available mapping of Parlay X interfaces onto OSA APIs we suggest a formal approach to specification of the OSA gateway behaviour. The approach may be useful in the design and implementation process of OSA gateway.

The paper is organized as follows. Section 2 describes some aspects of Parlay X deployment in IP-based multimedia networks and it discusses the open access to charging functions through Web Services and OSA Charging APIs. Section 3 presents the suggested mapping between the OSA Charging API and the Diameter protocol. Formal description of the finite state machine representing charging session handling and the Diameter peer state machine is provided in Section 4, where the state machines’ behavioural equivalence is proved. An example of Parlay X payment application is presented before concluding the paper.

2 Open Access to Charging Control

2.1 Parlay X Deployment in IMS Architecture
IMS is service control architecture intended to provide all types of multimedia services based on IP
As shown in Fig.1, Application Servers run 3rd party applications. Applications access network functions through Parlay X gateway attached to OSA gateway through OSA interfaces. The OSA gateway communicates with Serving Call Session Control Function (S-CSCF) which is responsible for user registration and session management relying on Session Initiation Protocol (SIP) signalling. The Home Subscriber Server (HSS) stores the user profiles. The access to user data in the HSS is based on Diameter signalling. The IMS online charging capable entities send charging information to Online Charging Function (OCF) via Diameter based interface [5]. The S-CSCF interacts with the OCF via IMS Gateway Function (IMS-GWF). The IMS offline charging capable entities collect charging information and send it to the Charging Data Function (CDF) via Diameter. The CDF constructs the Call Detail Records used for billing.

### 2.2 Open interfaces to charging functions

The Payment Web Service supports functions that allow applications to reserve, charge, and refund an account by volume or amount [6]. The OSA Charging APIs [7] provide methods for charging session control supporting crediting and debiting user accounts with specified amounts in number of monetary or non-monetary units directly or based on reservation.

Applications access charging functions by invoking operations of Parlay X Payment Web Service. The operation invocation opens an OSA charging session. The charging session handling is bounded by the invocation of OSA Charging interface methods. The mapping of Parlay X Payment operations onto Parlay/OSA Charging interface methods is defined in [8]. Each charging session opens a Diameter dialogue. No mapping is defined for the OSA Charging API onto Diameter protocol.

The OSA gateway needs to maintain two finite state machines; one representing the charging session handling as seen by applications and another one for the Diameter peer. An interoperability model requires both state machines to be synchronized i.e. to expose equivalent behaviour.

### 3 Mapping of OSA Charging Interfaces onto Diameter Protocol

The OSA methods `createChargingSession` and `createSplitChargingSession` of the IpChargingManager interface are used to create an instance of the IpChargingSession interface to handle the charging events related to the specified user or users. These methods are mapped onto the Start event defined in [9] by which the Diameter application initiates a connection with the peer as shown in Fig.2.

![Fig.2. Creation of OSA charging session](image)
charging server. It is mapped onto the Diameter Peer-Disc message.

![Diagram of Diameter Peer-Disc message](image)

**Fig.3. Indication of charging session termination**

The OSA release method of the IpChargingSession interface releases the session and it is mapped onto the Stop event [9] by which the Diameter application signals that the connection should be terminated as shown in Fig.4.

![Diagram of Stop event](image)

**Fig.4. Release of OSA charging session**

The OSA methods creditAmountReq, creditUnitReq, debitAmountReq, debitUnitReq, directCreditAmountReq, directCreditUnitReq, directDebitAmountReq, directDebitUnitReq, reserveAmountReq, reserveUnitReq, rateReq, extendLifeTimeReq, getAmountLeft, getLifeTimeLeft, and getUnitLeft of the IpChargingSession interface require actions to be taken in the network and these methods are mapped onto the Diameter message that has to be sent (noted as Send-Message in the Diameter peer state machine [9]). This Diameter message is Diameter Charging application specific [10]. The message is specific for the Diameter Charging applications [10]. It is Accounting Answer (ACA) for offline charging or Credit Control Answer (CCA) for online charging. Fig.5 shows the message sequence when the application credits an amount towards the reservation associated with the session in case of online charging.

![Diagram of message sequence](image)

**Fig.5. Crediting of user's account**

The failure handling behaviour reported to the application is locally configurable at the OSA gateway.

### 4 Charging Model Formal Specification

The formal specification of finite state machines as Labelled Transition Systems (LTS) allows proving the behavioural equivalence and hence the interoperability of OSA charging model and IMS charging model. This may be used for automatic generation of test cases during the OSA gateway verification.

We use the definition of LTS from [11] as a quadruple of a countable set of states ($S$), a countable set of elementary actions ($Act$), a set of transitions ($\rightarrow \subseteq S \times Act \times S$), and a set of initial states ($s_0 \in S$). The concept of weak bisimulation [12] is used to prove that both LTS expose equivalent behaviour.

#### 4.1 Charging Session Handling Described as LTS

The OSA charging session handling is described in [7] as a finite state machine of five states. The charging session does not exist in the Null state. The charging session is created in the Session Created state, but no reservations have been made. The application may perform direct debits and credits on the user's account and request rating. A reservation has been made for a certain maximum amount or certain maximum volume in the Amount Reserved state or the Volume Reserved state respectively. These are the states in which the
application may perform incremental debits/credits on the reserved amount/volume until either the application chooses to close the reservation or the reservation limit is reached, or the charging session is released. The application may also extend the reservation. In the Reservation Ended state, the reservation has been closed by the application, or the reservation limit has been reached. The charging session may remain active in order to carry out non-reservation related tasks such as direct credit or debit operations.

We present the OSA application view of charging session handling by LTS which is intentionally simplified. The actions in the formal definition correspond to the method invocations. To avoid exhausting repetition of similar expressions we use the following notations:

- commonList denotes any of the following methods:
  - directDebitAmountReq
  - directCreditAmountReq
  - directDebitUnitReq
  - directCreditUnitReq
  - rateReq

- amountList denotes any of the following methods:
  - getAmountLeft
  - debitAmountReq
  - creditAmountReq
  - reserveAmountReq

- volumeList denotes any of the following methods:
  - getUnitLeft
  - debitUnitReq
  - creditUnitReq
  - reserveUnitReq.

The transition to the Reservation Ended state is represented by the action reservationClosed. It is a result of application initiated reservation closing or reaching the original reservation limit.

By \( T_{\text{AppCSH}} = (S_{\text{AppCSH}}, Act_{\text{AppCSH}}, \rightarrow_{\text{AppCSH}}, s_0') \) we denote a LTS representing the OSA charging session handling where:

- \( S_{\text{AppCSH}} = \{ \text{Null, SessionCreated, AmountReserved, VolumeReserved, ReservationEnded} \} \)
- \( Act_{\text{AppCSH}} = \{ \text{createChargingSession, reserveAmountReq, reserveUnitReq, getAmountLeft, debitAmountReq, creditAmountReq, reserveAmountReq, directDebitAmountReq, directCreditAmountReq, rateReq, reservationClosed, release, sessionEnded} \} \)
- \( \rightarrow_{\text{AppCSH}} = \{ \text{Null createChargingSession SessionCreated, SessionCreated commonList SessionCreated, SessionCreated release Null, SessionCreated sessionEnded Null, SessionCreated reserveAmountReq AmountReserved, AmountReserved amountList AmountReserved, AmountReserved commonList AmountReserved, AmountReserved reservationClosed ReservationEnded, AmountReserved release Null, AmountReserved sessionEnded Null, SessionCreated reserveUnitReq VolumeReserved, VolumeReserved volumeList VolumeReserved, VolumeReserved reservationClosed ReservationEnded, VolumeReserved release Null, VolumeReserved sessionEnded Null, ReservationEnded commonList ReservationEnded, ReservationEnded release Null, ReservationEnded sessionEnded Null} \) \)
- \( s_0 = \{ \text{Null} \} \)

4.2 Diameter Peer State Machine Described as LTS

The Diameter peer state machine is defined in [9]. For methods invoked by the application, the OSA gateway behaves as a Diameter client agent, while for notifications the OSA gateway behaves as a Diameter server agent.

By \( T_{\text{Diameter}} = (S_{\text{Diameter}}, Act_{\text{Diameter}}, \rightarrow_{\text{Diameter}}, s_0) \) it is denoted a LTS representing a simplified Diameter peer state machine where

- \( s_0 = \{ \text{Closed} \} \)
- \( Act_{\text{Diameter}} = \{ \text{Start, Rcv-Credit-Ack, Rcv-CEA, Send-Message, Rcv-Message, Stop, Rcv-DPA, Peer-Disc} \} \)
- \( \rightarrow_{\text{Diameter}} = \{ \text{Closed Start Wait-Credit-Ack, Wait-Credit-Ack Rcv-Credit-Ack Wait-CEA, Wait-CEA Rcv-CEA Open, Open Send-Message Open, Open Rcv-Message Open, Open Stop Closing, Closing Rcv-DPA Closed, Open Peer-Disc Closed} \} \)
- \( s_0 = \{ \text{Closed} \} \)

4.3 Interoperability between OSA Charging model and IMS charging model

To prove the interoperability between charging models in OSA and IMS we have to prove that state machine representing the OSA charging session handling and the Diameter peer state machine expose equivalent behaviour. The behavioural equivalence is proved using the concept of weak bisimilarity [12].

Proposition 1: The labelled transition systems \( T_{\text{AppCSH}} \) and \( T_{\text{Diameter}} \) are weakly bisimilar.

Proof 1: To prove the bisimulation relation between two labelled transition systems, it has to be proved that there is a bisimulation relation between their states. The mapping between the OSA Charging interface methods and the Diameter messages defined in Section 3 shows the action’s similarity. By \( U \) it is denoted a relation between the states of \( T_{\text{AppCSH}} \) and \( T_{\text{Diameter}} \) where \( U = \{ (\text{Null, Closed}), (\text{SessionCreated, Open}) \} \). Table 1 presents the bisimulation relation between the states of \( T_{\text{AppCSH}} \) and \( T_{\text{Diameter}} \). Based on the bisimulation relation between the states of \( T_{\text{AppCSH}} \) and \( T_{\text{Diameter}} \) it is proved that both systems expose equivalent behaviour.
Table 1. Bisimilarity between OSA Charging Session Handling and Diameter Peer State Machine Transitions in $T_{AppCSH}$ Transitions in $T_{Diameter}$

Null createChargingSession Closed Start Wait-Conn-Ack,
SessionCreated Wait-Conn-Ack Wait-CEA,
AmountReserved Wait-CEA Rcv-CEA Open
SessionCreated commonList Open Send-Message Open,
SessionCreated Open Rcv-Message Open
AmountReserved Open Stop Closing,
AmountReserved Rcv-DPA Closed
SessionCreated release Null
AmountReserved Open Peer-Disc Closed
SessionCreated sessionEnded
AmountReserved Open Send-Message Open,
AmountReserved Open Rcv-Message Open,
ReservationEnded Open Send-Message Open,
ReservationEnded Open Rcv-Message Open,
ReservationEnded Open Peer-Disc Closed,
ReservationEnded release Null,
ReservationEnded Open Send-Message Open,
ReservationEnded Open Peer-Disc Closed
SessionCreated reserveUnitReq Open Send-Message Open,
VolumeReserved Open Rcv-Message Open,
VolumeReserved Open Send-Message Open,
VolumeReserved Open Rcv-Message Open,
VolumeReserved Open Peer-Disc Closed,
VolumeReserved Open Stop Closing,
VolumeReserved Rcv-DPA Closed
SessionCreated reserveAmountReq Open Send-Message Open,
AmountReserved Open Rcv-Message Open,
AmountReserved Open Send-Message Open,
AmountReserved Open Rcv-Message Open,
AmountReserved Open Stop Closing,
AmountReserved Rcv-DPA Closed
AmountReserved release Null,
AmountReserved Open Peer-Disc Closed
AmountReserved sessionEndedNull

5 An Example of Parlay X Payment Application

Let us consider an example application that uses the Parlay X Payment Web Service. The application grants a bonus of 1 free minute to users talking more than 10 minutes. We assume that the post-paid model of charging is applied.

The sequence diagram in Fig.6 shows the application reserving a specific volume (10 minutes) of credit in a user account (steps 1-13). The credit is increased (steps 14-19) and a bonus is granted (steps 20-25). The credit is deducted at a later point (steps 26-31). Finally the reservation is released (steps 32-38).

6 Conclusion

The paper investigates the interoperability of Parlay X Web Services payment model and the charging models in manageable IP-based multimedia networks. Using the available mapping of Parlay X Payment Web Services onto OSA Charging interfaces we provide a mapping of OSA Charging interfaces onto Diameter protocol. We suggest a formal approach to specification of OSA gateway which has to maintain two state machines representing the OSA charging session handling and the Diameter peer behaviour.

Because of OSA gateway 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.

Acknowledgments

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

References

Fig. 6 An example of Parlay X Payment application