

Analysis of State Models for Telecommunication Services

David E Vannucci & Hu E Hanrahan
Centre for Telecommunications Access and Services*
School of Electrical and Information Engineering
University of the Witwatersrand, Johannesburg, South Africa
{d.vannucci, h.hanrahan}@ee.wits.ac.za

Abstract—Telecommunication services are a profitable revenue stream for operators, and there is an increasing effort to offer these services through the Internet. Telecommunication services are currently being exposed to the Internet through a request response type manner that while ideal for simple services has performance issues for complex services. We examine an advanced service, a multimedia conference exposed through the Parlay X gateway, and examine how the lack of state and a state model affects operation of such a Web service. The Parlay state models are analysed and a mapping is proposed for a Web service state model to represent the multimedia conference.

I. INTRODUCTION

Telecommunication networks are evolving to offer the customer a greater choice in the chosen methods for using the network. The telecommunications network was previously a closed system operated and programmed solely by the network operator, but is now becoming an open system which allows third parties access to its functionality. This open access is driving the success of services.

Services represent a great potential source of revenue after that of standard voice traffic, and allows operators to realise the full potential of the network. Telecommunication operators are quickly moving to offer their services on the Internet.

The telecommunications service architecture can be viewed as a combination of Infrastructure and Services/Applications. The operator is continuously spending as small an amount as is possible on the network infrastructure, while making best use of these resources. Services allow the operator to get better leverage from the network equipment.

This paper reviews the current development of telecommunication services, with emphasis on the state models. This research forms part of a current research investigation into maintaining state within telecommunications Web gateways such as the Parlay X gateway. The call state models of selected telecommunication service architectures are investigated, to determine the requirements for a Web enabled service architecture.

In the following section the concept of the call model is examined, together with finite state machines. The Parlay call state models are discussed, so as to better understand how these relate to publishing services such as a multimedia conference on the Web. The state information available from a Web service point of view is then discussed by means

of a Web enabled multimedia conference service, and state models for this service are proposed.

A. The Call Model

Control of the network resources requires knowledge of the state of the resources and the current load on the network. The service architecture uses call state models to keep track of the progress of each session and the services that are used during the sessions. The call model represents all the essential features of a session from either the application's or network's point of view [1, pg. 39], and is a high level, technology independent abstraction of the call [2]. The call model describes the call state in terms of finite state machines. In [3] it was found that for any communication session, a call model can always be built to describe the logical entities involved and their inter-communication.

Many telecommunication services exposed to the Internet as Web services are of a simple message exchange type, such as sending a SMS, or sending a ringtone to a mobile phone. For these services a call state model is not necessary and neither is state information. Providing standard telecommunication services (like call waiting, conference calling, pay-per-call) require many messages to be exchanged, and control of the session is normally in place for the duration of the session. These services usually require the involved parties to implement a call model to interpret these messages based on the last known state of the session [4]. Through the implementation of a call state model a far richer set of service functionality becomes available, than that offered by a simple request response type [4]. Thus the call model can be thought of as the least common denominator supported by the call processing entities of the different domains [3].

Control of the network resources requires knowledge of the state of the resources and the current load on the network. The service architecture uses call state models to keep track of each session and the services being used during the sessions. Thus call state models are extremely important to the service application developer for successfully understanding and implementing services within the telecommunications network.

B. Finite State Machines

Finite State Machines (FSM) provide a powerful manner to describe the behaviour of reactive systems and their components [5], [6]. In [5] a FSM is defined as:

A State machine consists of states, events, transitions and actions. Each State has a (possibly

*The Centre is supported by Telkom SA Limited, Siemens Telecommunications and the THRIP Programme of the Department of Trade and Industry.

empty) State-entry and a State exit action that is executed upon State entry or State exit respectively. A transition has a source and a target State and is performed when the State machine is in the source State and the event associated with the transition occurs. For a transition t for event e between State A and State B, executing transition t (assuming the FSM is in State A and e occurred) would mean: (1) execute the exit action of State A, (2) execute the action associated with t , (3) execute the entry action of State B and (4) set State B as the current state.

Thus a FSM can be thought of as consisting of four main elements [7]:

- 1) States that define behaviour and can produce actions.
- 2) State transitions which are changes from one state to another.
- 3) Rules or conditions that are to be met to allow a state transition.
- 4) Events which are either externally or internally generated, which can possibly trigger rules and lead to state transitions.

II. PARLAY

The Open Service Architecture (OSA) and Parlay service architecture were standardised in order to facilitate the development of services by abstracting the network equipment and providing a common interface. This abstraction of the equipment made it possible to create services that communicated with the equipment in a standardised manner without the knowledge of the detailed, often proprietary, protocols and signalling of the equipment. The Parlay and Open Service Architecture (OSA) standards are converging into a common standard as a result of the similarities between the two service architectures.

OSA/Parlay defines a number of services, such as Call Control, Mobility, Charging. The one we will consider in this paper is the Call Control set of service APIs. Four Call Control APIs are defined:

- 1) Generic Call Control
- 2) Multiparty Call Control
- 3) Multimedia Call Control
- 4) Conference Call Control

The inheritance of the Call Control APIs are shown in figure 1. The Conference Call Control is not available in the OSA suite.

Parlay APIs are application-centric, i.e. only the parts of the call that are of specific interest to the application are defined within the call model. Application-centric call models are often simpler than network centric ones and the objects may be deleted if no further control of the call is required.

1) *Generic Call Control*: The Generic Call Control contains all the basic methods necessary to manage a two-party call. The Generic Call Control was designed with Capability Set 1 PSTN interworking in mind, and as such does not support third-party initiated sessions. The API facilitates only basic services and provision for more than two parties is not supported.

The Parlay Call Control Working group together with JAIN, ETSI and other involved parties has focussed on

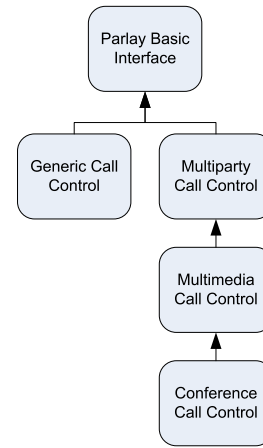


Fig. 1. OSA/Parlay Call Control Inheritance [1, pg. 163]

the Multiparty Call Control and multimedia call control APIs, and these are greatly improved from the Generic Call Control. Thus the joint call control group decided that the Multiparty Call Control API is to be considered as the future base call control and technical work on the Generic Call Control has been suspended.

2) *Multiparty Call Control*: The Multiparty Call Control API extends the capabilities of the Generic Call Control API. The Multiparty Call Control supports third party initiated sessions, and has a richer call model. The Multiparty Call Control does not inherit from the Generic Call Control API as the decision was made to avoid having changes to the Multiparty Call Control effecting the Generic Call Control.

The Multiparty Call Control Service is represented by three interfaces: `IPMultiPartyCallControlManager`, `IPMultiPartyCall`, `IPCallLeg`.

The finite state machines of the `IPMultiPartyCallControlManager` and `IPMultiPartyCall` are shown in figure 2. There are two FSMs for the `IPCallLeg`, originating and terminating, as shown in figure 3.

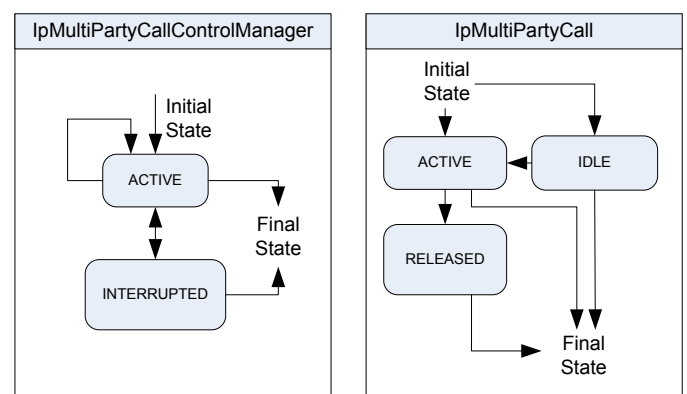


Fig. 2. Multiparty call control FSMs

3) *Multimedia Call Control*: The Multimedia Call Control service enhances the functionality of the MultiParty Call Control Service with multimedia capabilities [8, pg. 15], adding control to sessions with multiple, and different medias. The Multiparty Call Control requires participants to use the same media, for example the same voice codecs, and the same video codecs [1, pg. 174]. To facilitate the handling of a multimedia call, the Parlay Group introduced

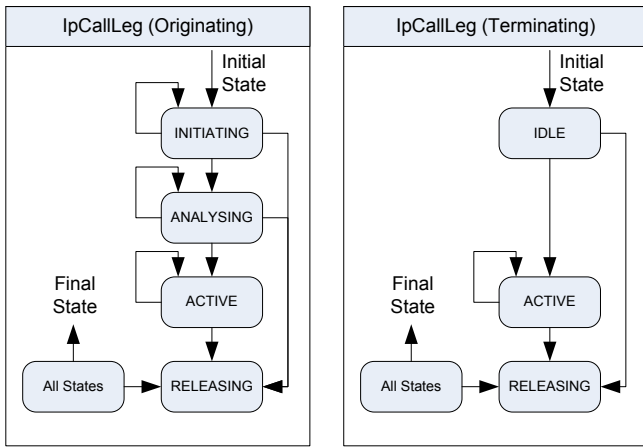


Fig. 3. Multiparty IpCallLeg call control FSMs

the concept of a bidirectional media stream associated with a call leg, negotiated between the terminals involved with the call. The Multimedia Call Control Service consists of four interfaces to services provided by the network: IpMultiMediaCallControlManager, IpMultiMediaCall, IpMultiMediaCallLeg and IpMultiMediaStream. The finite state machines for the interfaces in the Multimedia Call Control API inherit from the interfaces in the Multiparty Call Control API and are the same. The IpMultiMediaStream interface does not have a finite state machine.

4) *Conference Call Control*: The Conference Call Control Service extends the functionality of the Multimedia Call Control service, and provides specialised conferencing functionality and resources. The Conference Call Control service allows the controlling application to manipulate subconferences within a conference, allowing the conference chair to add and remove parties. The Conference Call Control API does not introduce any new FSMs.

III. EXAMPLE APPLICATION: A MULTIMEDIA CONFERENCE

In a Web environment there is less need for an extremely complex call model, rather varying levels of complexity are required depending on the application developed and the intended user group. Thus an evaluation is required to determine the level of state information available to the Web service and the call models which are possible with this information.

In this example we consider the case of a multimedia conference call being configured from the Internet. In the case of this multimedia conference call Web service the progression of the conference can be separated into a number of steps:

- 1) Create the conference
- 2) Invite participants
- 3) Add media for participants
- 4) Delete media for participants
- 5) Disconnect participants
- 6) End conference

These steps are discussed in the following sections.

1) *Create the Conference*: The conference is activated by the Parlay X Web Conference logic receiving a request to start a new conference. Starting the conference initiates a

new Multimedia Call Control manager object in the Parlay gateway to handle the conference objects for the duration of the conference, the manager in turn creates a call object, as shown in the sequence diagram in figure 4. As the createConferenceRequest is issued by the Parlay X Web service (Application Logic) one would expect that there would be a change of state for the conference from null to a initialising state, initial, as the Parlay X Web service would be the first to update its conference model. However as the Parlay X Web service is stateless this does not occur, and the Web service has to specifically request a conference info update by means of a getConferenceInfoRequest message sent to the Parlay gateway. The Parlay gateway Multimedia Conferencing object is able to update its status of the conference immediately. Once the conference has been created, the conference status is set to Initial.

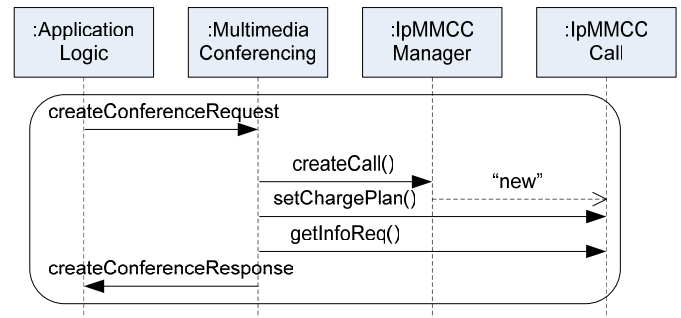


Fig. 4. Create Conference

Note that while the new message is asynchronous (denoted by an open arrow) all the remaining messages are synchronous. The messages are synchronous to ensure the Web service request response model is maintained.

2) *Invite Participants*: Details of the participants are added by the Web service so that these participants can be invited, the first participant being the conference chair.

The Parlay Multimedia conferencing object would create a new call leg listener to receive the notification of when the participant picks up the phone and the Call object is requested to create the new call leg for the participant.

The Multimedia conferencing object would also update its version of the Participant status to invited, and when notification is received from the network that the call has been answered, the status would be changed to connected.

Note that the Web service would not be aware that the participant has answered or not answered the phone until it specifically queries the Parlay gateway. This requires the Web service to be continuously polling the Parlay gateway (by means of the getParticipantInfoRequest request) for status information.

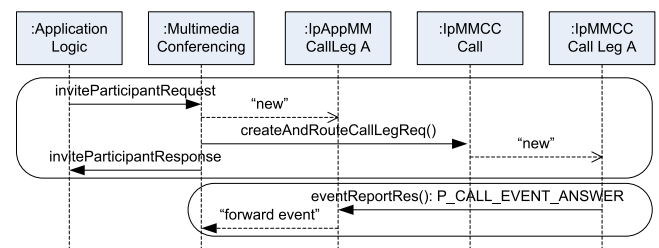


Fig. 5. Invite Participant

3) *Add Media*: In addition to Voice there are three other media types supported by the Parlay X Web conferencing service: Video, Chat and Data [9]. All of these media types can be unidirectional or bidirectional. The request is issued to the participants terminal and if the terminal is capable of supporting this request the media stream is created. The status of the participant is then updated in the Multimedia Conference object. This causes a discrepancy in the state information between the Web service and the Parlay gateway, as the Web service has to poll the Parlay gateway to determine if the request to add the media was successful.

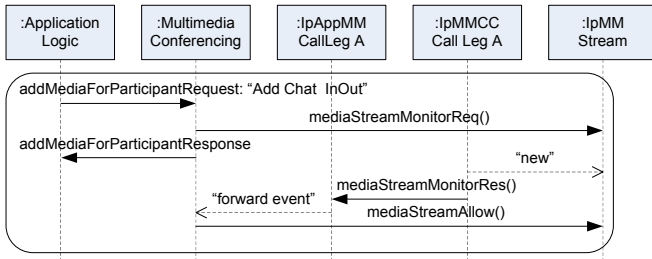


Fig. 6. Add Media

4) *Delete Media*: When a media stream is to be disconnected, the Web service would issue a delete media request. The Parlay gateway Multimedia Conferencing object would then update its conference status. The Web service would have to query the Parlay gateway to determine if the request was successful.

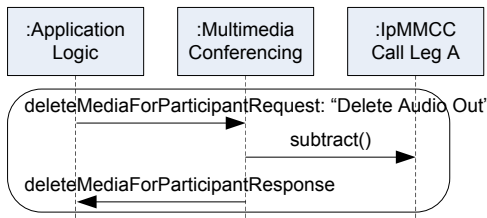


Fig. 7. Delete Media

5) *Disconnect Participants*: Participants may be removed from the conference by the Web service issuing a `disconnectParticipantRequest`. This results in the Multimedia conference object in the Parlay gateway issuing a `release` to the call leg, and removing the event listener on that leg. In [10] the Parlay X Web server only updates the state of the participant when it specifically requests the participant info.

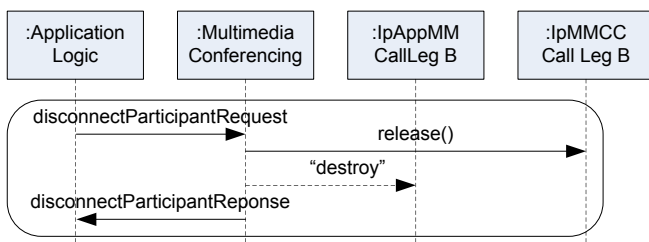


Fig. 8. Disconnect Participants

6) *End Conference*: There are two manners in which to end a conference, firstly by the Web service issuing an `endConferenceRequest`, and secondly by the conference owner disconnecting the terminal. The first case is shown in figure 9. If the conference chair disconnects, the Web service would incorrectly report the conference as active until an update of the conference status is requested.

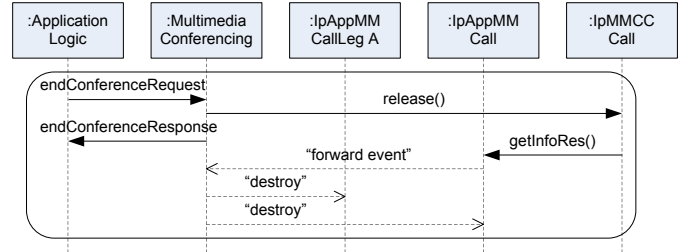


Fig. 9. End Conference

A. State in the Parlay X Gateway

The state in the Parlay X Web service gateway is updated using the `getConferenceInfo` request and the `getParticipantInfoRequest`. These two methods are the only ways to query the current state from the Parlay gateway.

Since Parlay X Web services are stateless, the Parlay gateway cannot initiate an uninvited notification to the Parlay X Web service. Thus the Web service would always assume the last known state is correct. In addition issues arise when events happen within the network, without the request of the Web service, for example a participant disconnects. This information is not available to the Web service until a request for the participant information is sent.

Thus the state within the Parlay Gateway can be thought of as extremely detailed within the call control objects and less detailed within the Multimedia Conferencing object, as illustrated by figure 10.

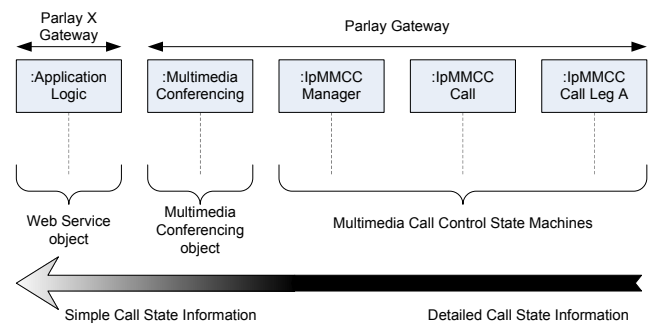


Fig. 10. State Information in Parlay X and Parlay

The Web service state follows that of the Multimedia Conferencing object but is delayed until an update is requested. In addition it may be the case that the Web service would not be interested in certain aspects of the participants details such as the media types in use. The level of detail required for the Web service is of interest as it is possible that a call state model can be presented which could be even

more simplified than that of the information stored by the Parlay multimedia conference object. If one assumes that the Web Service state is the same as that stored by the Parlay multimedia conference object then from the example above [9] two finite state machines can be created for the conference Web Service:

- The participant finite state machine
- The conference finite state machine

For the state of the participants the finite state machine can be simplified from a full call leg model as in figure 3 to that in figure 11. Initiating and analysing would be contained within the initial state, when the call leg is active this would correspond to the connected state, and the call leg releasing state would correspond to the disconnected state as shown in figure 11. Also this participant model would be the same regardless of whether the call leg is originating or terminating.

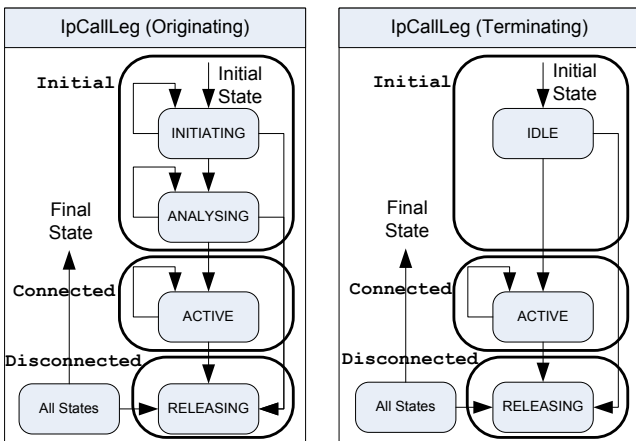


Fig. 11. Web Service States mapped to IpCallLeg call control FSMs

For the state of the conference the finite state machine would contain three states: Initial, active and terminated. The conference state machine can be mapped to the IpMultiPartyCall object. In the IpMultiPartyCall, idle would correspond to the initial state, active to active, and the released would correspond to terminated as shown in figure 12.

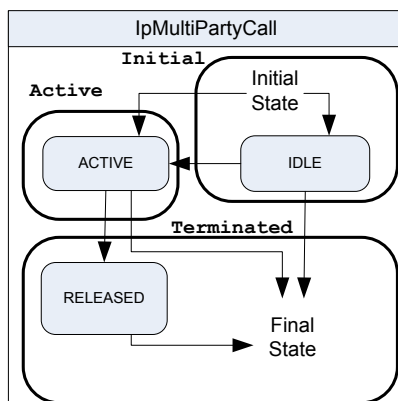


Fig. 12. Web Service States mapped to IpCall call control FSM

Thus the Web conference states could be described as in figure 13.

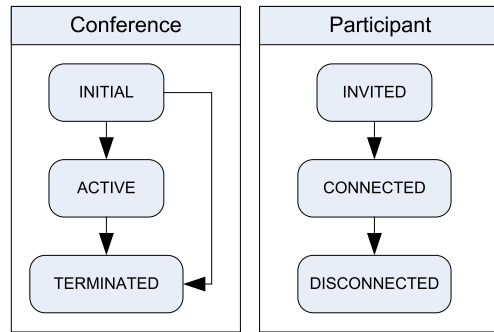


Fig. 13. Web Service States

IV. CONCLUSION

This paper has investigated the need for call state models for telecommunication Web services. Using the Parlay APIs an example of a Web service for multimedia call conferencing was considered to better understand the dynamics and operation of a telecommunications Web service. Some of the problems associated with stateless Web Services were identified and illustrated in the example. Finite state machine models were created for the example to describe the conference states and the states of the participants.

REFERENCES

- [1] R. Jain, J.-L. Bakker, and F. Anjum. *Programming Converged Networks*. John Wiley & Sons, Washington, USA, first edition, 2005.
- [2] M. Graf. An Introduction to the Java Telephony API (JTAPI). Technical report, IBM Research Division, March 2000. <http://www.zurich.ibm.com/csc/distribsys/j323/jtapi-tutorial.pdf>.
- [3] J. Dobrowolski, M. Grech, S. Qutub, M. Unmehopa, and K. Vemuri. Call Model For IP Telephony. Technical Report Internet Draft, IETF, June 1999.
- [4] J. Dobrowolski, W. Montgomery, K. Vemuri, J. Voelker, and A. Brusilovsky. IN Technology for Internet Telephony Enhancements. Technical Report Internet Draft, IPTeL Working Group, December 1999. <http://quimby.gnus.org/internet-drafts/draft-dobrowolski-iptel-in-00.txt>.
- [5] J. van Gurp and J. Bosch. On the Implementation of Finite State Machines. In *Proceedings of the 3rd Annual IASTED International Conference Software Engineering and Applications*, Scottsdale, Arizona, USA, 6-8 October 1999.
- [6] David Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3):231-274, June 1987.
- [7] A. Kaul, N. Varshney, R. K. Koyana, S. P. Krishna, U. Shankara, and V. Jaiswal. Finite State Machine (FSM) Framework for SIP Extensions. Technical Report Internet Draft, Internet Engineering Task Force, March 2003. <http://www.softarmor.com/wgdb/docs/draft-kaul-sip-fsm-framework-00.txt>.
- [8] ETSI. Part 4: Call Control; Sub-part 4: Multi-Media Call Control SCF (Parlay 5). Technical report, European Telecommunications Standards Institute, The Parlay Group, April 2005.
- [9] Parlay ETSI. Open Service Access (OSA); Parlay X Web Services; Part 12: Multimedia Conference. Technical report, European Telecommunications Standards Institute, The Parlay Group, March 2005. http://www.parlay.org/specs/docs/es_20239112v010101p.zip.
- [10] Parlay ETSI. Open Service Access (OSA); Mapping of Parlay X Web Services to Parlay/OSA APIs; Part 12: Multimedia Conference Mapping. Technical report, European Telecommunications Standards Institute, The Parlay Group, January 2005. <http://portal.etsi.org/docbox/TISPAN/Open/OSA/ParlayX/ParlayX.Mapping/2005.04/DTR-TISPAN-01021-12v003.doc>.