Modular Design of Call Control Layer in Telephony Software

Ilija Basicevic

University of Novi Sad, Faculty of Technical Sciences
Novi Sad, 21000, Serbia

Abstract

An important property of a telephony system is the call control model on which it is based. It is noted that many call control models in the past, especially those in PSTN/ISDN networks follow centralized model. For such a model, typical is significant coupling of modules belonging to different services with the basic call control module which is aware of all active telephony features in the system. Although sometimes based on distributed model, VoIP call control models still manifest some of the listed problems of their predecessors. In this paper we present a fully distributed model which exhibits minimal coupling of modules belonging to different services and a simple basic call control module. The model is based on taxonomies of call control services which are presented in the paper. Also, the implementation of several typical services is described.

Keywords: *Call Control, telephony services, Voice over IP, VoIP session transfer, VoIP session redirection*

1. Introduction

Call control model can be described as a formal representation (and design) of a distributed software system for telephony communications. Typically, there is a network consisting of infrastructure and endpoints. This network can be represented as a graph. Infrastructure nodes are responsible for routing. Endpoint nodes are nodes that have only one adjacent node and are usually responsible for end users' access to services of telephony network. Call control model specifies the design and mutual interaction of software modules that are responsible for call processing.

The aim of this paper is twofold. One is to present the specific call control model that is developed here. An important issue with respect to that is the issue of modular development of telephone features. For rather long time, designers have strived to developed fully modular features, decoupled from the code of the so called "basic call control" and other features. The other aim of this paper is to focus on the feature management module, which is an important part of the call control layer.

Section 2 describes related work, section 3 presents two taxonomies of call processing features that are elaborated in this paper, and the manner in which some of the features are implemented. Section 4 presents the implementation of a feature that belongs to the class of network based services. Section 5 contains concluding remarks.

2. Related Work

There has been a lot of research in the area of feature management and call control models. Influential call control models that are used in circuit switched telephony have been published by Telecommunication Standardization Sector of the International Telecommunications Union (ITU-T). We mention here Q.71 [1] model used in PSTN/ISDN networks and Intelligent Network (IN) model [2]. In Q.71 model, each new feature that is introduced to the call processing system leaves its "fingerprint" in the basic call control module. The IN is the first model in software industry that features systematic definition and operational adoption of service orientation [3]. The IN architectural stack clearly identifies several well defined layers of service with distinct responsibilities and roles. The next step in the telecommunications industry has been the development of object oriented application programming interfaces (API). We mention here Parlay, the 3GPP Open Service Architecture and Java APIs for Integrated Networks (JAIN). A simplified version of Parlay/OSA has been later extended with support for Web-services, and XML [4] resulting in Parlay X [5], which is a de-facto standard Web-services API today.

Distributed Feature Composition [6] has been an attempt to develop a highly distributed and decoupled model. The advent of Voice over IP (VoIP) telephony brought H.323[7] and Session Initiation Protocol (SIP) [8] models. There has been continuous work on improvement of those models as in WSIP[9] and in compositional control of IP[10]. SIP protocol is based on two-step and three-step transactions, while in compositional control an idempotent signaling protocol based on unilateral descriptions is proposed. WSIP is an integration of two concepts, SIP and Web Services. The idea behind WSIP is separation of service integration signaling. Thus we have a three tier stack of service integration, signaling and media

transmission. In more recent papers, call control is researched as a part of collaborative streaming applications [11]. IP Multimedia Subsystem (IMS), which is based on combination of IN concepts and application of Internet protocols, most importantly SIP and Diameter[12], appeared in 2004. As of today, IMS is considered a global standard for unified service control platform converging fixed, mobile, and cable IP networks [13].

Although telephony end points of today are way simpler than switches of PSTN/ISDN/IN networks, the most important aspects of feature management problem are present in both types of systems.

3. Two Taxonomies of Call Processing Features

The concept presented in this paper is implemented in the framework [14], but for convenience of readers, some details required for understanding the paper are repeated here. EndUser class models the end user of telephony endpoint, and contains the typical telephony endpoint interface. SignalingDevice is a class that interfaces the underlying telephony protocol stack (SIP, for example). Feature class is the parent of all classes that model telephony features (e.g. Session, CallWaiting, SessionRedirect etc.). Session class models the basic call feature with first party call control interface. P3Session also models the basic call feature but with the third party call control interface. The important methods of Session are: Invite, AcceptSession, Disconnect, ModifySession, and the callbacks for messages from the remote peer: OnAccepted, OnDisconnect, OnModifySession. The Invite message from the remote peer is handled in the EndUser object - at this moment the local Session object does not exist. FeatureMng is the feature manager class which dispatches received messages to feature modules. It is the responsibility of this class to determine which active features, and in which order will process the received message. EndPoint models the session terminal available to end users for utilizing network services. RoutingPoint is the infrastructure node responsible for routing of messages. RoutingPointExt is the infrastructure mode extended with the software modules of network side applications.

We have introduced simple taxonomy of features, based on analyses of standards and implementation results. Each feature is either primitive, derivative or composite. Basic service session has two classes, one with the interface for first and the other with the interface for third party call control. Basic session with the interface for first party call control is considered primitive feature and the root of this taxonomy. Derivative features are session transfer, session redirect, session waiting, etc. Composite features are basic session with interface for third party call control and conference.

The relationship between composite and primitive feature is similar to "has" relationship (aggregation) in Unified Modeling Language (UML).

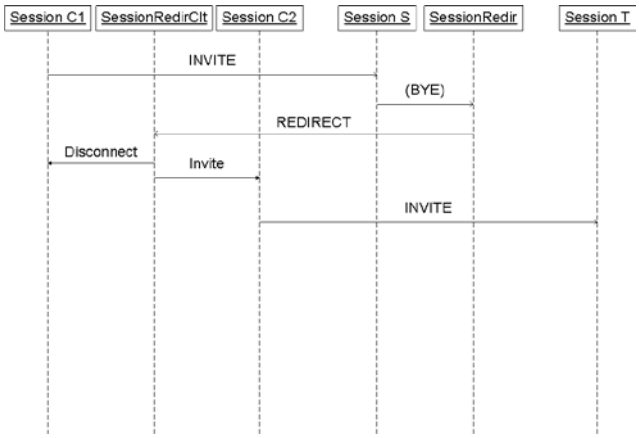

Figure 1: Session redirect MSC

Each feature's intelligence knows about itself and its primitive elements' features. Each feature registers at the feature manager for notification about call state changes. In notification, manager follows the rule that active primitive features are notified before derivative and composite ones. For example basic call (Session) is notified about session change before call waiting feature. For sessions, this rule reduces to the following: the basic call is the first feature to be notified about any session state changes.

Certain derivative features have read access to basic call state. This is used for checking preconditions of certain session control operations. No feature has full (read and write) access to another feature data. Feature manager fully recognizes only primitive features. All derivative and composite features are considered by feature manager only as instances of the generic class Feature. Regarding unwanted feature interaction, it is assumed that end user will not simultaneously activate features that result in unwanted feature interaction. Although this is a very simplifying hypothesis, we consider unwanted feature interaction to be out the scope of this paper.

Typical feature communicates the EndUser object (for receiving end user commands), the SignalingDevice object (for sending signaling messages to remote peers), the feature manager (for receiving commands from remote peers), but there is a category of derivative features that communicate the EndUser only for the reasons of feature activation and configuration. During operation phase there is no interaction with local end user in such features. Those features manipulate the call automatically, without the local user explicitly taking part in the call control. Example is call redirection.

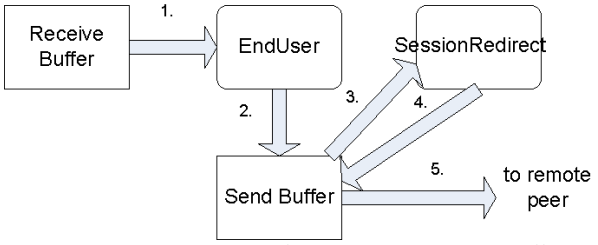

Figure 2: Chain processing of received messages (send buffer)

There is another taxonomy of telephony features, depending on the programming model. In the first class there are features that are realized as peer-to-peer (symmetrical) modules. Such a feature is basic call. Both sides in the session communication are the same. In the second class there are features that are realized using client-server model (asymmetrical). An example is session redirection feature, see Fig. 1. There is SessionRedirect class implementing server side that receives INVITE message (sent by Session object) and responds with REDIRECT message, and there is SessionRedirectClt class that receives REDIRECT message, closes the first session (Session object that sent the INVITE) and opens a new one by sending INVITE message to address referred to in REDIRECT message. The SessionRedirectClt class implements the client side of the feature. While SessionRedirect extends the Session class, SessionRedirectClt inherits the Feature class. An important aspect of operation of this feature is the chain processing of received messages. At the redirect server side, end user module is the first to process received INVITE. In case end user has already been engaged in a call, BYE message with a reason that the end user is already in call will be sent. The next in chain is session redirect server, which processes the BYE message, removes it from the send buffer and replaces it with REDIRECT message. The chain processing is presented in Fig 2. It can be presented with the following pseudo code.

```
while (sendbuffer not empty){
    msg = get_message(sendbuffer)
//message is deleted from sendbuffer, iterator
//moves to next
    if(relevant_to_operation_of_the_feature(msg))
        process (msg)
}
```

Processing of the message typically includes placing a different message in the send buffer. When the last feature in the chain finishes processing, messages in the send buffer are actually sent over the network to remote peer. The order of features in the chain is the result of the order in which they were activated. This order is very important for the feature interaction, but as noted earlier, we assume that it is the responsibility of the system's end user.

In order to sustain a relatively small number of basic message types, we have introduced the following message information field to messages: source feature type. Thus we can use the same message type (ACCEPT) for confirming session establishment and session transfer, or for example the same message type (BYE) for session tear down both in first party and third party call control. In those cases, ACCEPT will carry either Session' or SessionTransfer' identificator as source feature type and BYE will carry either Session' or P3Session' identificator in that message information field. Another reason for introducing this message information field is the chain processing of received messages, thus each feature knows which feature reacted before it in the chain, and placed a message in the send buffer. For example, if BYE is from SessionTransferClt, SessionRedirect feature will ignore it. But it will react to BYE which was placed into the send buffer by Session feature.

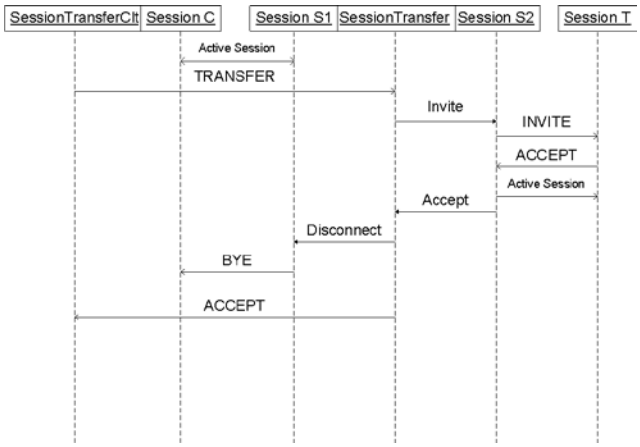

Figure 3: Session Transfer MSC

Another client-server based feature is transfer. There is SessionTransferClt class that implements the client side. This class contains the TransferSession method. When the user asks for transfer, EndUser object invokes the TransferSession method of SessionTransferClt object. In the next step, the SessionTransferClt's method sends TRANSFER message to SessionTransfer object at the remote peer. Two transfer feature objects are positioned at two sides of an active session. Upon receiving the message, SessionTransfer object instantiates another Session object, and invokes its Invite method. An INVITE is sent to the target point of transfer operation, see Fig. 3 We have assumed here that the target responds with ACCEPT. This message is processed by two feature modules: Session and SessionTransfer. As we already noted FeatureManager first dispatches the message to Session (as a primitive feature) and then to SessionTransfer (as a derivative feature).

SessionTransfer object disconnects the session to transfer client by sending BYE (it invokes Disconnect method of Session object after it gets the handle of the session object from feature manager). Immediately afterwards it confirms the successful transfer by sending ACCEPT to SessionTransferClt object at the remote side.

The example of session redirection feature (and session transfer) shows the value of modular approach. This feature can be implemented without the SessionRedirectClt class by placing its logic in Session class. (Also the transfer feature can be completely realized in Session class without introducing SessionTransfer and SessionTransferClt). In that way, the Session class grows more and more complex. It becomes the module that "knows everything" about call processing. The consequence is that call control layer becomes much more error prone. In contrast, modular development allows for gradual increase of functionality. The call control layer is easier to test and debug. The feature interactions are controlled in a more straight forward manner.

The third party call interface of basic session, realized in P3Session class, provides the following operations: Establish, and Terminate. The Establish method, that establishes basic session between two remote points is based on transfer and monitoring features. Monitoring is based on publish/subscribe event notification mechanism.

4. Network based services

The services mentioned in the Section 3 are end point based. There is no service specific processing in infrastructure points. In this section we will give an example of implementation of network based service. Line hunt is another ISDN supplementary service. It belongs to the group of network side services. In the framework such services are usually implemented by inheriting RoutingPointExt class. Thus a new class RoutingPointExt1 has been implemented. This class contains a linked list of hunt mappings. The mapping contains session layer address that is mapped, and network layer address and port it is mapped to. Since in line hunt one session layer address is mapped to a group of network layer addresses, the logical relationship of one line hunt mapping is realized as a group of list elements, all containing the same value of session layer address field.

The dynamics of the line hunt operation is realized in the following manner. The routing of the protocol message is intercepted. Typically, the router thread of SignallingDevice class passes the received message to the RouteProtMessage of RoutingPoint class, which then inspects the routing table and forwards the message according to the appropriate route in the table.

In case of the line hunt operation, router thread passes the received message to RoutingPointExt1 class. In case of INVITE and ACCEPT messages, the routing for line hunt is different from aforementioned general routing procedure. The processing of INVITE sets the flag of line hunt for that session layer address to active. A copy of INVITE message is sent to all addresses the target address of INVITE is mapped to. The first received ACCEPT from one of those addresses sets the flag to false, also a BYE is sent to all the other addresses that INVITE has been sent to.

5. Conclusions

During the decades long evolution of telephony software, there have been several approaches to design of call control layer. However, the majority of them featured a strongly centralized approach where basic call control module with inclusion of new features becomes a "know all" module. The legacy of PSTN/ISDN networks of fixed telephony has been the Q.71 model. With the appearance of the IN network, ITU-T invested in an approach where service plane and basic call control plane would be strongly separated. This paper presents strongly modular design of call control layer where inclusion of new features is possible without modification of existing ones, especially having in mind the basic call control module. Telephony features in this model are implemented as asymmetrical, client server modules, while basic call control module is implemented as symmetrical, peer-to-peer module. Implementations of session transfer and session redirect telephony features described in the paper show this separation between processing in the basic call control and other features. The implementation of line hunt service is given as an example of network based service.

References

[1] ITU-T Q.71 - ISDN Circuit Mode Switched Bearer Services, (1993) International Telecommunication Union
[2] ITU-T Recommendation Q.1200, Q-Series Intelligent Network Recommendation Structure, (1993) International Telecommunication Union
[3] Tiziana Margaria, (2007), Service is in the Eyes of Beholder, Computer Magazine, vol 40, No 11, DOI:10.1109/MC.2007.398
[4] Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation, 26 November 2008
[5] ETSI OSA PArlay x 3.0 Specifications, European Telecommunications Standards Institute and The Parlay Group, 2007
[6] Jackson M., Zave P., (1998), Distributed Feature Composition: A Virtual Architecture for Telecommunications Services, IEEE Transactions On Software Engineering, vol. 24, no. 10

[7] Rosenberg J., Schulzrinne H., Camarillo G., Johnston A., Peterson J. Sparks R., Handley M. Schooler E., (2002), Session Initiation Protocol, RFC 3261, Internet Engineering Task Force
[8] ITU-T H.323, Visual Telephone Systems and Equipment for Local Area Networks Which Provide A Non-Guaranteed Quality Of Service, (1996), International Telecommunication Union
[9] Liu F., Chou W., Li L., Li J., (2004), WSIP - Web Service SIP Endpoint for Converged Multimedia/Multimodal communication over IP, IEEE International Conference on Web services (ICWS 04), pp.690
[10] Zave P., Cheung E., (2006), Compositional Control of IP Media, International Conference On Emerging Networking Experiments And Technologies, Lisboa, Portugal
[11] Kahmann V., Brandt J., Wolf L., (2006), Collaborative Streaming and Dynamic Scenarios, Communications of ACM, vol 49, no 11.
[12] Calhoun P., Loughney J., Guttman E., Zorn G., Arkko J., (2003), Diameter Base Protocol, RFC 3588, Internet Engineering Task Force
[13] Magedanz T., Blum N., Dutkowski S., (2007), Evolution of SOA concepts in Telecommunications, Computer Magazine, vol 40, No 11, 2007, DOI:10.1109/MC.2007.384
[14] Basicevic I., (2009), Object-Oriented Framework for Development of Telephony Applications, Fourth International Conference on Digital Telecommunications, Colmar, France

Ilija Basicevic received his BSc, MSc, and PhD degrees from the University of Novi Sad in 1998, 2001 and 2009 respectively. Currently, he is assistant professor at the same university. His research interests are communication systems and computer security. He has published more than 30 papers. Ilija is member of ACM and IEEE.