Parlay API 中 GCC SCF 向 SIP 协议的映射研究

樊自甫,万晓榆
FAN Zi-fu,WAN Xiao-yu

重庆邮电大学 下一代网络应用技术研究所,重庆 400065
Chongqing University of Posts and Telecommunications,Chongqing 400065,China
E-mail:fzfboy@sina.com.cn

FAN Zi-fu,WAN Xiao-yu.Research of GCC SCF mapping to SIP in parlay API.Computer Engineering and Applications,2007,43(17):133-136.

Abstract: The 3rd party service creation parlay API-based is the most opening service provision style in next generation network,the 3rd party application programming transfers the lower layer network resource by the SCF interface defined in Parlay API mapping to lower layer network protocol (e.g.SIP,INAP etc.,).Research the problem of GCC SCF mapping to SIP protocol, introduce the call model of Parlay API and SIP,then bring forward the model of GCC SCF mapping to SIP,and design the specific program realization flow.Analyze number translate service's process finally.
Key words: NGN;parlay API;GCC SCF;SIP;mapping

摘 要:基于 Parlay API 的第三方业务开发是下一代网络(NGN)中最为开放的业务提供方式,第三方应用程序通过 Parlay API 中 SCF 接口向底层网络协议(如 SIP、INAP 等)的映射来调用底层网络资源。论文重点对 Parlay API 中向 SIP 协议的映射问题进行研究,首先介绍了 Parlay API 和 SIP 的呼叫模型,接着提出了 GCC SCF 向 SIP 协议的映射模型,并设计出映射模型的程序实现流程,最后以号码翻译业务为例分析了该映射模型的工作过程。
关键词:下一代网络;Parlay API;GCC SC;SIP;映射
文章编号:1002-8331(2007)17-133-04 **文献标识码**:A **中图分类号**:TP393

1 引言

下一代网络(NGN)是一种业务驱动网络,存在着多种业务提供方式,其中基于 API 的业务提供方式能够快速引入第三方业务,而成为一种最能体现 NGN 业务开放性特征的业务提供方式。为此,NGN 中基于 Parlay API 的第三方业务提供方案成为了当前研究的一个热点,其中一个关键的内容是如何实现 Parlay API 到底层网络协议的映射问题。本文就 Parlay API 中 GCC(Generic Call Control,通用呼叫控制)SCF 到 SIP 协议的映射问题,作了较为深入的分析,提出了一种实现 GCC SCF 到 SIP 协议的映射模型,并对模型中需要考虑的问题作了一定的探讨,供研究 Parlay API 向 SIP 协议映射的广大同行参考。

2 Parlay API 与 SIP 协议中的呼叫模型

由于 Parlay API 和 SIP 位于 NGN 体系结构中的不同层次,它们分别从不同的角度来定义和反映呼叫。Parlay API 中与呼叫相关的概念主要有 Call 和 CallLeg。Call 是指在多个通信实体间建立的信令关系,在呼叫控制 SCF 中对应为由 CallSessionID 标识的呼叫对象。CallLeg 表示一个 Call 与一个地址(即参与方身份)之间的联系,在呼叫控制 SCF 中对应为由 CallLegSessionID 标识的呼叫腿对象和由<CallSessionID,CallLegSessionID>标识的呼叫对象与呼叫腿对象之间的联系。一个 Call 可能包含多个参与方,每个参与方通过一个 CallLeg 把自己的身份或地址信息与该 Call 相关联。

SIP 中与呼叫相关的概念主要有 Call 和 Dialog。Call 是指在对等 SIP UA(useragent)之间建立的联系,由 Call-ID 唯一标识。Dialog 是指在两个 SIP UA 之间持续一定时间的对等 SIP 关系,由 Dialog-ID 来标识,具体包括 Call-ID、localtag(本地标签)和 remotetag(远端标签)等三部分内容。一个 Call 由一个或多个 Dialog 组成,同一 Call 中的 Dialog 拥有相同的 Call-ID,并通过 remotetag 和 localtag 参数限定该 Call 的参与者。

3 GCC SCF 向 SIP 协议的映射模型

在设计 GCC SCF 向 SIP 协议的映射模型前,首先来讨论一下通过 Parlay API 实现第三方业务时需要解决的问题。

从应用层面看,第三方的一个应用有可能使用多个 SCF。这样一来,要保证应用和各个 SCF 之间通信的正常进行,一方面要维护应用和各 SCF 的对应关系,这可以通过应用和框架以及 SCF 之间的签约来完成。另一方面,各 SCF 可能同时支持

基金项目:重庆市教委科学技术研究项目(No.KJ050512,No.KJ050513,No.KJ060514);重庆市科委自然科学基金项目(No.CSTC 2005BB2054);重庆邮电大学青年教师基金项目(No.A2005-26)。
作者简介:樊自甫(1977-),男,讲师,主要研究方向为下一代网络技术;万晓榆(1963-),男,教授,博士,硕士生导师,主要研究方向为下一代网络技术。

多个应用的调用，所以在 SCF 中必须能够区分底层的消息应该提交给哪个应用，这种对应关系的维护可以通过在 SCF 中为每个使用它的应用创建一个 SM(业务管理器)对象来实现，可能发生的呼叫由该 SM 进一步创建其他的对象来完成，即在 SCF 中某个 SM 对象及其创建的一系列对象都可以明确地服务于某个应用。从上面的分析可以看出，对每个 SCF 的实现可以通过一个应用程序来完成。

从 SCF 到网络底层协议映射的层次来看，在 SCF 中可能需要将各种方法参数映射到多种协议(如 SIP,INAP 等)，因此在 SCF 各方法的实现中，与各种协议的实现代码之间应尽量保持为一种松耦合关系，同时，考虑到同一种协议可能为多个 SCF 提供服务，因此协议栈应该尽可能地独立。

基于以上考虑，本文在实现 GCC SCF 功能时，采用了将整个 GCC SCF 作为一个应用程序来实现，对 GCC 中的各个接口类程序由相应的类来实现。总体的设计思路是第三方的应用逻辑由底层协议栈的相关事件触发，然后由 SCF 将事件通知给第三方应用，由第三方应用来确定如何处理这些事件，事件的处理是通过第三方应用调用 SCF 中的相关方法来完成。其软件模块结构如图 1 所示。

图 1 GCC SCF 软件架构

图 1 中，Corba 接口模块完成 SCF 与应用和框架的通信，可以使用相关的 Corba 产品来生成 Corba 规范中的客户端的 stub 和服务器端的 skeleton。GCC 主模块完成自身的初始化过程，如完成到框架(FW)接口的注册、完成和应用之间在线业务协定的签署、初始化和启动其他模块的工作、实现 GCC 中定义的各个接口类及其方法等。SIP 协议栈模块是相对独立的模块，主要完成 SIP 消息到底层网络的发送和接收，并实现对 SIP 消息的解析。事件标准解析模块负责将 SIP 协议栈接收到的 SIP 消息解析为应用设置的事件通知标准的数据结构，从而实现将 SIP 消息转换为标准事件，使得 SCF 实例能够顺利完成事件的匹配工作。另外，软件架构中还需要维护 SIP 服务器的工作模式信息，即对于当前 GCC 实例而言，SIP 服务器是处于何种模式(如 Proxy 服务器模式、Redirect 服务器模式、UA 模式、B2BUA 模式和第三方控制器模式等)，具体模式的选择根据应用向 SCF 传递的请求来确定。在 GCC 实例需要发送 SIP 消息时，GCC 的实现对象首先设定 SIP 服务器的模式，将需要发送的内容交给 SIP 服务器，由 SIP 服务器生成具体的 SIP 消息，然后调用 SIP 协议栈发送出去。因此，SIP 服务器模块中主要包含实现 SIP UA(用户代理)，Proxy(代理)，Redirect(重定向模式)，第三方控制器，B2BUA 等模式的相关类。

4 GCC SCF 向 SIP 映射的程序设计

整个 GCC 的实现程序基于 VOCAL 的 SIP 协议栈和 SIP UA、B2BUA 代码。GCC SCF 部分主要有 IpCallControlManager 类和 IpCall 类，还有 EventCriteriaParser(事件标准解析)类，以及与 SIP 有关的各个类。在 GCC SCF 中，各个类之间的关系如图 2 所示。

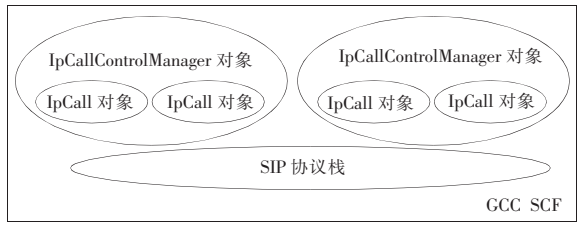

图 2 GCC SCF 中各个类对象的关系

图中可以看出，GCC SCF 中可能包含多个 IpCallControlManager 对象，分别对应于不同的第三方的业务逻辑程序，每个 IpCallControlManager 对象作为一个 SCF 实例的管理者，还可以创建多个 IpCall 对象，分别对应于同一个应用的不同呼叫。SIP 协议栈是大家共用的。

在 GCC 程序中，main()函数，主要完成创建 IpCallControlManager 对象之前的一系列和 Parlay 框架(framework)接口及应用之间的交互过程，然后再启动 SIP 协议栈。在 SIP 协议栈启动后，将会自动创建其收发消息的工作线程。对于某个应用来说，此时已有了一个为其提供服务的 IpCallControlManager 对象。IpCallControlManager 类的声明如下所示：

```
Class IpCallControlManager :public IpService
{
    Public:
        IpCallControlManager();
        ~IpCallControlManager();
// 实现 Parlay API 规范中的各个方法
        TpCallIdentifier createCall(IpAppCallRef appCall);
        TpAssignmentID enableCallNotification (IpAppCallControlManagerRef appCallControlManager,TpCallEventCriteria eventCriteria);
        void disableCallNotification(TpAssignmentID assignmentID);
        TpAssignmentID setCallLoadControl (TpDuration duration, TpCallLoadControlMechanism mechanism,TpCallTreatment treatment, TpAddressRange addressRange);
        void changeCallNotification(TpAssignmentID assignmentID, TpCallEventCriteria eventCriteria);
        TpCallEventCriteriaResultSet getCriteria();
        Void run ();//完成 IpCallControlManager 对象的主要的内部操作
    Private:
        /* 接收来自 IpServiceInstanceLifeManager 对象中 createServiceManager()方法所传递而来的与应用相关的属性，由此完成应用所要求的具体 SM 对象的创建 */
        TpClientAppID myClientAppID;
        TpServicePropertyList myServicePropertyList;
        TpServiceInstanceID myServiceInstanceID;
        TpCallEventCriteria myCallEventCriteria;//保存应用所设置的事件通知标准，用于对应用逻辑的触发
        IpAppCallControlManagerRef myAppCallControlManagerRef;//保存与该 SM 对应的应用侧的通信对象
        TpCallIdentifierList myCallIdentifierList;// 维护当前应用中已有的 Call 对象
};
```

IpCallControlManager 类对于第三方应用通过 GCC SCF 来实现对 SIP 协议的操作至关重要，是整个呼叫处理的管理者。其中，run()方法的主要工作流程如下：从 SIP 协议栈的消息队列中提取消息，调用 EventCriteriaParser 类来完成底层网络事

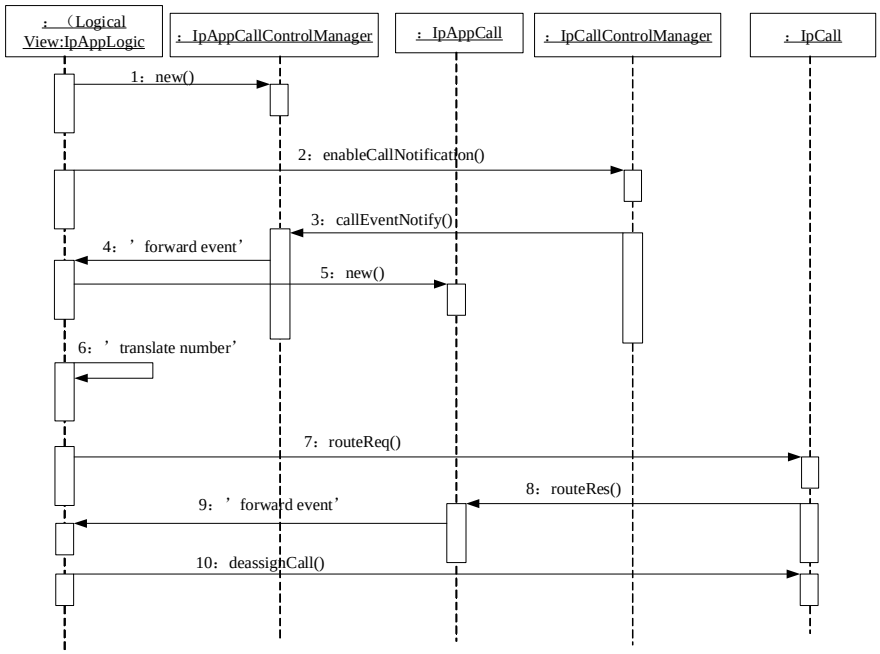

图 3 基于 GCC SCF 的号码翻译业务流程

件和应用所设置的事件通知标准之间的匹配,若匹配则调用 IpAppCallControlManagerRef.callEventNotify()方法,通知应用逻辑有其感兴趣的事件发生。然后,应用调用该管理器中的方法来创建 Call 对象,以完成对呼叫的处理。呼叫处理到 SIP 协议的映射是在设定 SIP 服务器的模式后, 调用 VOCAL 中的函数如 proxy_main()、B2BUA_main()、Redirect_main()、UA_main()等来完成。在使用这些代码时,由于其本身对实现 SIP 服务器的各个模式是各自独立的程序,即分别有其自己的 main()函数,此时需要将这些 main()函数的名称改写一下。

在 GCC SCF 实现中,另一个重要的类为 IpCall 类,该类的每一个对象对应作用于该 SCF(第三方应用)的每一次呼叫,因此此类主要完成对呼叫的路由、释放等工作,同时完成前面所述的 Parlay API 和 SIP 消息中针对 SIP 服务器不同模式所进行的不同的方法和参数映射。

5 号码翻译业务实现流程举例

号码翻译业务(NTS)使用对外统一号码(逻辑号码)来免费接入用户的咨询电话,然后由该业务流程前转到某个真正的物理号码上,业务逻辑类似于 800 服务。在利用 Parlay API 中 GCC SCF 来实现该业务的流程如图 3 所示。

号码翻译业务的触发, 可以看成是对号码字冠的匹配结果。从整个业务来看,GCC SCF 中的 SIP 服务器应该工作于 B2BUA 模式,因为它首先需要将用户发起的呼叫进行终结,然后再向目的端发起一个新的呼叫。具体流程如下:

(1)在应用逻辑方面,首先需要创建一个 IpAppCallControlManager 对象,以便让 GCC SCF 有回调的接口。

(2)应用逻辑调用 GCC SCF 中 IpCallControlManager 对象(此时该对象已经完成初始化, 已经可以代表该应用逻辑与运营商的底层网络进行通信)的 enableCallNotification()方法,设置触发该应用的事件,假设此业务的业务码为 315,则对应的事件通知标准如下:

```
TpCallEventCriteria myCallEventCriteria
{
    TpAddressRange DestinationAddress={
        Plan=P_ADDRESS_PLAN_E164;地址方案为 E.164 地址
        AddrString="315*";以 315 开头的号码
        Name=null;没有定义相关的名称
        SubAddressString=null;没有附加号码
        };
    TpAddressRange OriginatingAddress=null;不关心主叫的信息
    TpCallEventName CallEventName=
P_EVENT_GCCS_ADDRESS_COLLECTED_EVENT OR
P_EVENT_GCCS_ADDRESS_ANALYSED_EVENT; 指定需要报告的事件为号码分析事件和号码分析完成事件
    TpCallNotificationType CallNotificationType=P_TERMINATING;表明以上需要报告的事件与被叫相关,即需要从被叫信息中提取
    TpCallMonitorMode MonitorMode=P_CALL_MONITOR_MODE_NOTIFY;指定事件通知模式为 NOTIFY,即当有满足事件通知标准的事件发生时,立即通知给应用,而不用打断呼叫的正常进行
}
```

(3)在设定相关事件通知标准后,当有到此应用的呼叫到达时,GCC SCF 中的 IpCallControlManager 对象会调用应用逻辑中 IpAppCallControlManager 对象中的 callEventNotify()方法来告诉应用,通知应用它所感兴趣的事件已经发生。

(4)应用逻辑从 IpAppCallControlManager 对象获知事件内容。

(5)应用逻辑创建一个新的 IpAppCall 对象来处理此次呼叫相关的具体进展情况。

(6)应用逻辑将逻辑号码转换为物理号码(某电话机号码或某移动手机号码)。

(7)应用逻辑调用 GCC SCF 中的 IpCallControlManager 对象来创建一个新的 IpCall 对象,然后使用已翻译的号码作为新的呼叫目的端号码。根据以上分析,号码翻译业务中 SIP 服务器采用 B2BUA 模式时,在 routeReq()方法中,Parlay API 中的 CallSessionID、CallLegSessionID 与 SIP 中的 Call-ID、LocalTag、RemoteTag 之间的映射关系如表 1 所示。

表 1 NTS 业务中 SIP 参数与 GCC SCF 参数之间的映射

SIP	头域	OSA/PARLAY API	Leg	CALL
SIP	call-ID(1)			callSessionID
Dialog	From 头域中的 local tag		CallLegSessionID(1),主叫 Call Leg(1)对象	
#1	To 头域中的 remote tag			
	Request-URI	originalTargetAddress		Call 对象
SIP	call-ID			
Dialog	From 头域中的 local tag			
#2	To 头域中的 remote tag		CallLegSessionID(2),	
	Request-URI	targetAddress(经过翻译后的号码)	目的端 Call Leg(2)对象	

由表 1 可以看出,假设 Dialog #1 为拨打 NTS 业务的用户所创建的 SIP Dialog,则 Dialog #1 的 call-ID 映射为 GCC SCF 中 callSessionID,From 头中的 local tag 映射为 GCC SCF 中的主叫方的 callLegSessionID,Request-URI 映射到 GCC SCF 中的 originalTargetAddress。经过应用逻辑的处理后,应用程序调用 GCC SCF 中的 routeReq()方法完成向已经过转换的号码发起呼叫,即在 routeReq()方法中实现在 SIP 服务器 B2BUA 模式下的 INVITE 邀请,此时将会创建一个新的 SIP Dialog,即表中的 Dialog #2。在 Dialog #2 中 SIP 消息的 call-ID 与 Dialog #1 中的 call-Id 是不同的,它们分别由 GCC SCF 中的 SIP 服务器(Server)和客户(Client)产生。同时,Dialog #2 中 To 字段的 remote tag 映射为到真实目的端的 CallLegSessionID,但从 GCC SCF 的角度来看,这两个 SIP Dialog 属于同一个 Call 对象,因而 callSessionID 并没有发生变化。所以,在 IpCall 对象中必须要维护属于该呼叫对象的不同 SIP Dialog 之间的关联关系。

(8)IpCall 对象返回与新呼叫相关的一些进展信息。使用的方法为 routeRes(),该方法会实时地将底层 SIP 协议的响应代码映射为该方法中的参数 TpCallReport eventReport 中。其内容可能为以下信息:

TpCallReport eventReport=

{

TpCallMonitorMode myMonitorMode =P_CALL_MONITOR_MODE_NOTIFY;监控模式为 NOTIFY

TpDateAndTime CallEventTime;事件发生的日期和时间

TpCallReportType CallReportType =P_CALL_REPORT_ANSWER;被叫已经应答

TpCallAdditionalReportInfo AdditionalReportInfo=NULL; 对呼叫报告的补充

}

此处,SIP 响应消息和呼叫报告类型之间有一定的对应关系,如 TpCallReportType 和 SIP 响应码之间的对应关系如表 2 所示。

表 2 TpCallReportType 与 SIP 响应码的对应关系

TpCallReportType	SIP 响应代码
P_CALL_REPORT_PROGRESS	100
P_CALL_REPORT_ALERTING	180
P_CALL_REPORT_ANSWER	200
P_CALL_REPORT_BUSY	600
P_CALL_REPORT_NO_ANSWER	486
P_CALL_REPORT_REDIRECTED	181
P_CALL_REPORT_QUEUED	182
P_CALL_REPORT_NOT_REACHABLE	604

(9)应用逻辑从 IpAppCall 对象获取呼叫进展信息(如呼叫成功)。

(10)应用完成本次呼叫后,解除与呼叫对象之间的关联,也即释放对 GCC SCF 中的 IpCall 对象的引用。

6 结语

在 Parlay GCC SCF 向 SIP 协议的映射中,两者的呼叫模型以及层次都有所不同,映射中需要从概念、方法和参数等几个方面考虑。在设计 GCC SCF 向 SIP 映射的软件架构中,我们将整个 GCC SCF 的功能通过一个应用程序来实现,对 GCC 中定义的各个 API 接口类由相应的类程序来实现,第三方应用逻辑由底层协议栈的相关事件触发后,由 SCF 将事件通知给第三方应用,由第三方应用来确定如何处理这些事件。本文所提方案中,第三方应用需要与其自己的 SCF(SCS)一起才能工作,主要原因是 SIP 服务器模式是根据应用的实际功能来设定的,无法由第三方应用实现 SIP 服务器模式的自动判别。因而,Parlay API 中的 GCC SCF 没能完全屏蔽底层网络协议细节,如果 SCF 能够根据应用需求自动地转换 SIP 服务器的模式,才可以做到 SCF 与第三方应用无关的特性。

(收稿日期:2006 年 10 月)

参考文献:

[1] Lu Tian,Eng M.A CORBA-based interface-centric approach to signaling for IP-based telephony services[EB/OL].[2005-05].http://www.sce.carleton.ca/netmanage/papers/LuThesis.pdf.

[2] OMG.CORBA/TC interworking and SCCP Inter-ORB protocol specification[EB/OL].http://www.omg.org/technology/documents/formal/corba_tc_interworking_and_sccp_i.htm.

[3] ETSI ES 202 915-1 V1.2.1.Open Service Access(OSA);Application Programming Interface(API);Part 1:Overview(Parlay 4),2003-08.

[4] Open Service Access (OSA);Application Programming Interface (API);Part 4:Call Control;Sub-part 2:ETSI ES 202 915-4-2 V1.2.1[S].Generic Call Control SCF(Parlay 4),2003-08.

[5] Glitho R H.Poulin,A.A high level service creation environment for parlay in a SIP environment[C]//IEEE International Conference on Volume 4,28 April-2 May 2002:2008-2013.

[6] Guo Le-shen,Chen Jun-liang.Next generation intelligent network based on CORBA and agent technologies[C]//Communication Technology Proceedings,International Conference on Volume 2,9-11 April 2003:1562-1565.

[7] 万晓榆,姚平香,张洪,等.下一代网络的业务生成技术[M].北京:北京邮电大学出版社,2005.

[8] 樊自甫,万晓榆.NGN 基于 Parlay API 业务生成接口的研究与设计[J].计算机工程与应用,2006,42(3):127-130.