基于JAIN SIP的用户代理软件设计

陈东郎,朱翠涛
(中南民族大学 电子信息工程学院,湖北 武汉 430073)

摘 要:会话初始化协议(SIP)是下一代网络中应用层的信令控制协议。JAIN SIP是用于实现SIP应用而提供的一套标准Java接口。提出了一个基于JAIN SIP的用户代理软件框架,包括图形用户界面(GUI)、SIP消息处理模块、媒体处理模块和SIP协议栈,并对各个功能模块进行了详细分析。用JAIN SIP来开发基于SIP协议的应用不仅可以提高开发效率,而且将使应用程序获得较高的可靠性和较好的移植性。

关键词:SIP协议;JAIN SIP;用户代理
中图分类号:TP393.03 **文献标识码**:A **文章编号**:1673-629X(2006)12-0130-03

Software Design of User Agent Based on JAIN SIP

CHEN Dong-lang, ZHU Cui-tao
(*Coll. of Electronics & Info. Eng., Central South University of Nationalities, Wuhan 430073, China*)

Abstract: The session initiation protocol is the session controlling protocol of application layer of NGN(next generation network). JAIN SIP, is the Java-standard interface to a sip signaling stack for implementing SIP application. Present a framework of user agent based on the architecture and implementation mechanism of JAIN SIP, analyze the function of each module which include the graphical user interfaces, processing of SIP information, processing of multimedia information and the architecture of JAIN SIP. Developing SIP-based applications with JAIN SIP will improve the development efficiency, and the applications will gain high reliability and portability.

Key words: SIP protocol; JAIN SIP; user agent

0 引 言

SIP协议[1]是IETF制订的应用层的信令控制协议,用来建立、修改和终止两个或多个参与者的会话。它定义了两个要素:SIP用户代理和SIP网络服务器。IETF的SIP工作组对SIP协议的基本实体、体系结构、协议操作以及消息格式等都作了详细的规范,SIPPING工作组则对基于SIP协议的各种应用包括网络电话、多媒体服务等进行研究和制定规范。JAIN(Java Advanced Intelligent Network) APIs是由JCP组织推动开发的一套基于Java APIs和面向对象技术的标准接口,主要用于在Java平台上快速开发下一代电信产品业务。JAIN SIP API[2]是SIP协议标准化的Java接口,该技术规范由Sun和dynamicsoft公司合作研发,完全基于IETF的SIP协议规范(RFC3261),是一个Java技术的协议栈,该协议栈提供了各种SIP消息的编程接口,用户根据提供的这些接口来实现应用编程开发。

随着VoIP技术的进一步发展和视频会议系统应用的逐渐普及,国内外对基于SIP协议的应用开发越来越广泛,各个公司和社团都纷纷推出了自己的SIP用户代理产品,包括SIP软件电话和SIP硬件电话。SIP软件电话如Xten Networks公司的X-Lite、EyeP Media公司的EyeP Phone、Java开源社区的sip-communicator、时曦科技的Bol SIPPhone等,硬件电话如3Com公司的SIP Phone,SIP Etherphone等。这些用户代理产品均支持标准的协议,同时还提供了会议功能。文中基于JAIN SIP协议栈,提出了一个用户代理的软件框架,并对各个模块进行了详细分析。

收稿日期:2006-04-01
基金项目:国家自然科学基金资助项目(30370393)
作者简介:陈东郎(1982-),男,浙江永康人,硕士研究生,研究方向为软交换与网络融合;朱翠涛,博士,副教授,研究方向为软交换与网络融合技术。

1 用户代理软件结构

用户代理是一个用户与其它用户代理或服务器通信的终端。采用SIP协议作为信令协议,该用户代理可以运行在各种系统平台上,支持音频、视频、文本和电子白板等媒体类型。基于JAIN SIP协议栈的用户代理[3]软件结构如图1所示。

从图1可知,用户代理的内部结构大致可以分为4个部分:图形用户界面(GUI)、SIP消息处理模块、媒体处理模块和JAIN SIP协议栈。其中JAIN SIP协议栈又分SIP包和SDP包两部分,分别负责SIP消息和媒体流的解析和发送。另外,通过图形用户界面与用户的直接交互控制其余模块的操作,并为它们分配资源。

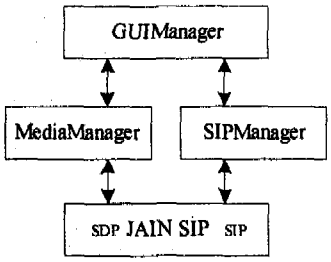

图 1　用户代理软件结构

1.1　图形用户界面(GUI)

图形用户界面模块提供所有实现 SIP 用户代理功能所需的标准界面,包括主窗口、登录注册窗口、配置窗口、聊天窗口、视音频窗口等。在窗口中包含各种常用的组件如按钮、复选框、标签、文本框、列表框等,这些可见组件都是窗口小部件,通过事件驱动来调用协议栈的消息发送机制和实现状态的迁移转换等。

主窗口是整个用户代理程序的入口点,首先定义一个 GUIManager 用户代理主类,该类定义了 main 方法[4]。在该类的构造函数中,通过调用其它模块的类方法来完成程序的初始化。初始化过程为:定义 Configuration 类完成基本参数的配置;定义 MediaManager 类的 detectSupportedCodecs()方法完成媒体设备的检测和初始化;定义 SipManager 类的 start()方法完成 SIP 协议栈的初始化。GUIManager 主类在 initComponents()方法中定义了各个组件和参数,同时在各个组件上增加事件的监听者,用来响应用户的请求并调用相应的类来实现该功能。

1.2　SIP 消息处理模块

SIP 消息处理模块主要负责呼叫控制信令的处理,完成用户的各种请求、响应和超时事件。它构建于 SIP 协议栈提供的 SIP 包之上,继承了 SIP 协议栈提供的 SipListener 接口。

SipListener 接口定义了 3 个方法:processRequest(RequestEvent requestEvent), processResponse (ResponseEvent responseEvent), processTimeout (TimeoutEvent timeoutEvent)。在 SIP 消息处理模块中定义 MessageListener 类,继承 SipListener 接口,将其中的 3 个方法实现。在 MessageListener 类中分别通过请求事件得到服务器端事务和响应事件得到客户端事务来判断消息的类型,如果是请求消息则在处理请求方法中进一部判断请求消息的类型,分别是 INVITE, ACK, BYE, CANCEL, MESSAGE, SUBSCRIBE, NOTIFY,再根据不同的请求消息类型分发给相应的功能单元进行处理;如果是响应消息则在处理响应的方法中判断响应的消息类型,分别是 OK, TRYING, RINGING, SUBSCRIBE ACCEPTED, NOT_FOUND, UNAUTHORIZED 等,OK 响应又可进一步判断为 REGISTER, INVITE, BYE, CANCEL, MESSAGE, SUBSCRIBE 响应,再根据不同的响应消息类型分发给相应的功能单元进行处理。同样的,对超时事件也通过判断超时事件类型,并交给相应的功能单元进行处理。以上 MessageListener 类所有定义的消息处理方法都交给 MessageProcessor 类来实现。在 Message Listener 类中同时还定义了协议栈的初始化 start()方法,初始化过程为:

(1) JAIN SIP 应用程序调用 createListener()方法创建出一个全局唯一的 SipListener 对象;

(2) SipListener 通过 getInstance()方法创 SipFactory 对象,并通过其创建 SipStack 对象;

(3) SipStack 对象调用 creatSipProvider() 方法生成 SipProvider 对象,并将其私有 SipStack 交由 SipProvider 管理。

当 SipProvider 成功建立,SipListener 就对其进行注册,建立消息/事件机制,完成初始化。

在 MessageListener 类之上定义了 MessengerManager 类,该类是 SIP 消息处理模块的核心类,在该类中定义了 call(String contactURI), answerCall(String caller), addContact(String contact), register()等方法,这些方法直接对应用户的操作事件,完成用户的呼叫管理、增加删除联系人、注册到代理服务器等功能,所有 GUIManager 主类中定义的监听事件的处理都通过 MessengerManager 类来实现。

1.3　媒体处理模块

媒体处理模块主要负责媒体信息的采集、解析和发送。在端系统的混合模式中,媒体处理模块还需要对音频数据进行混合分发。媒体处理模块构建于 JAIN SIP 协议栈提供的 SDP 包之上,独立于 SIP 消息处理模块,它通过 Java 多媒体框架(JMF)[5]来实现。

在媒体处理模块中定义了 MediaManager 类,该类是媒体处理模块的核心类,我们在类中定义了 detectSupportedCodecs(), startReceiving(), startTransmitting(), prepareMediaSession(String incomingSdpBody), findCorrespondingSdpFormat(String jmfFormat)等方法。在 detectSupportedCodecs()方法中完成媒体设备的检测和初始化,初始化过程为:

(1) 通过 CaptureDeviceManager 类的 getDeviceList(null)方法得到音频采集设备列表,并由设备列表的 elementAt(i)方法选取第 i 个采集设备作为本系统采集设备,然后通过采集设备的 getLocator()方法获取一个媒体定位器 locator;

(2) 由媒体定位器、IP 地址、端口、视音频编码格式作为参数构造 RtpTransmit 传输类的对象;

(3) 在 RtpTransmit 传输类中,通过 javax.media.Manager 类的 createDataSource(locator)方法产生一个数据源 ds,然后把数据源 ds 作为参数,通过 javax.media.Manager 类的 createProcess(ds)方法产生一个处理器 processor。再用 StateListener 类的 waitForState(processor, Processor.Configured)方法等待处理器配置好;

(4) 通过 Process 类的 getTrackControls()方法为媒体流中的每一个轨道得到一个控制器,并确保至少有一个可用的轨道。再创建 ContentDescriptor 类的对象 cd,通过

Process 类的 setContentDescriptor(cd)设置输出的内容描述为 RAW_RTP,限定合法的 RTP 格式;

(5) 最后对每一个轨道,选择一种 RTP 支持的传输格式,完成初始化。

用户代理在完成了媒体的初始化后就可以创建 RTP 会话,开始媒体流的发射与接收了。媒体流的发射与接收由该类的 startTransmitting(),startReceiving()这两个方法来实现。

1.4 JAIN SIP 协议栈

JAIN SIP 对 SIP 通信实体的实现采用模块化处理,每个实体都由 SipStack,SipProvider 和 SipListener 构成。整个架构以事件为基础,采用了监听者(Listener)/提供者(Provider)的事件模型,其中事件封装了 SIP 实体所接受到的 SIP 消息。主要是向上层提供 SIP 消息的封装、解析、监听和发送功能。JAIN SIP 协议栈结构如图 2 所示。

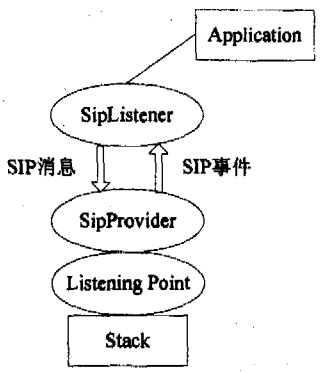

图 2 JAIN SIP 体系结构

SipStack 是 SIP 协议栈,是整个 JAIN SIP 的核心实现部分,所有操作都在这里实现,包括地址产生、消息编解码、消息事件的生成、会话和传输的建立,以及地址的解析等。SipStack 是 SipListener 和 SipProvider 的基础,其实现的 SIP 服务由 SipProvider 统一向 SipListener 提供。

SipProvider 是一个事件的提供者,通过 Listening Point 封装了对 Stack 进行操作的方法,将 SipStack 接受到的 SIP 消息加工成事件交给 SipListener 处理,并将上层应用生成的 SIP 消息交由 SipStack 处理。SipProvider 提供了客户端事务(Client Transaction)和服务器端事务(Server Transaction)的创建方法,对有状态的请求/应答消息提供了良好的支持。SipProvider 同时支持无状态的请求/应答消息,这是 SIP 的默认消息类型。一个应用程序中可以有多个 SipProvider,每个对应一个 Stack。

SipListener 负责管理一个或多个 SipProvider。SipProvider 的行为依赖于应用程序的业务逻辑。当一个 SipProvider 建立之后,将被注册到一个 SipListener;应用程序访问某个 SipProvider,就通过 SipListener 向该 SipProvider 发送消息;如果 SipProvider 收到了其他 SipProvider 的 SIP 消息,需要将其传递给应用程序进行解析,此时它将产生一个事件,SipListener 理解事件的含义并能根据不同的事件触发相应的响应。

2 结束语

SIP 协议作为下一代网络的核心协议,以其易于扩展、开放的业务生成环境、对移动性的支持等特点而具有广泛的应用前景,而基于 Java 技术制定的 SIP 协议栈规范 JAIN SIP 将对 SIP 消息的处理转化为对消息和事件的处理,极大地提高了开发的效率。对基于 JAIN SIP 的用户代理进行了设计分析,介绍了各个模块的功能及软件设计。本设计的实现对开发基于 JAIN SIP 协议栈的增值业务将起到一定的促进作用。

参考文献:

[1] Rosenberg J, Schulzrnne H, Camarillo G, et al. RFC3261 SIP: Session Initiation Protocol[S]. IETF, 2002.

[2] SunMicrosystems. JAIN SIP Tutorial[EB/OL]. 2004 - 06. http://java.sun.com/products/jain/.

[3] AIN - SIP - APPLET - PHONE[EB/OL]. 2003 - 11. http://snad.ncsl.nist.gov/proj/iptel/.

[4] Lewis J, Loftus W. Java 程序设计基础[M]. 第 3 版. 王锦全译. 北京:清华大学出版社, 2004.

[5] Java Media Framework[EB/OL]. 2004 - 02 - 10. http://java.sun.com/products/java2media/jmf/index.html.

(上接第 129 页)

服务器的处理负荷,在容错方面,在节点随机失效的情况下也提高了系统的鲁棒性。最后,系统可扩展方面更加的简单。

参考文献:

[1] Foster I, Kesselman C, Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations[J]. International J Supercomputer Applications, 2001, 5(3): 200 - 222.

[2] Foster I, Kesselman C. The Grid: Blueprint for a New Computing Infrastructure[M]. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.

[3] Chervenak A, Foster I, Kesselman C, et al. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets[J]. Journal of Network and Computer Applications, 2001, 23: 187 - 200.

[4] Allcock W. GridFTP Protocol Specification[EB/OL]. 2003 - 03. GGF GridFTP Working Group Document, http://www.globus.org/alliance/publications/papers/GFD - R.0201.pdf.

[5] Watts D, Strogatz S. Collective dynamics of smallworld networks[J]. Nature, 1998, 393: 440 - 442.