文章编号:1001 - 9081(2005)02 - 0276 - 03

利用 Java 技术实现 SIP 通信

杨 鹏[1],赵 博[1],王 琨[2],周利华[1]
(1. 西安电子科技大学 多媒体研究所,陕西 西安 710071;
2. 西安电子科技大学 计算机网络与信息安全教育部重点实验室,陕西 西安 710071)
(yangpengp@163.com)

摘 要:SIP 作为应用层的会话控制协议,具有简单、扩展性和扩容性好等优点。在简要介绍 SIP 协议的基础上,详细讨论了 SUN 公司用于实现 SIP 通信的 JAIN SIP 开发架构,并利用 Java 语言以 JAIN SIP 为核心,描述了具体实现 SIP 通信中各通信实体的基本方法,为 SIP 通信搭建了简单的模型。

关键词:SIP; JAINSIP; Java; SIP 通信; 通信实体
中图分类号: TP393.04 **文献标识码**:A

Using Java technology to implement the SIP communication

YANG Peng[1], ZHAO Bo[1], WANG Kun[2], ZHOU Li-hua[1]
(1. *Institute of Multimedia Technology, Xidian University, Xi' an Shaanxi* 710071, *China;*
2. *Key Lab of Institute of Computer Network and Information Security Ministry of Education, Xidian University, Xi' an Shaanxi* 710071, *China)*

Abstract: As the session controlling protocol of application-layer, SIP has the features of simple, expansibility and dilatancibility. At the basis of simple introduction of SIP protocol, the JAIN SIP exploring construction for the fulfillment of SIP communication of SUN Co was discussed in detail. To use Java language and take JAIN SIP as the core, all kinds of communication entity basic method in the fulfillment of SIP communication were described and simple model for SIP communication was built.

Key words: SIP; JAINSIP; Java; SIP communication; communication entity

会话初始化协议(Session Initiation Protocol,SIP)最近几年受到了极大的关注,将 SIP 作为第三代移动通信的信令协议提供 IP 多媒体服务的决定,使得 SIP 成为又一个技术热点。SIP 将蜂窝系统与 Internet 应用领域融合在一起,使用户能够把传统的 Internet 服务,比如 E-mail、Web 以及多媒体和即时消息等新服务结合起来。

1 SIP 介绍

会话发起协议(Session Initiation Protocol,SIP)是由 IETF 提出的 IP 电话信令协议,用于生成、修改和终结一个或多个参与者之间的会话,在 IP 网上实现端到端的通信。SIP 协议源于 HTTP 协议(超文本传输协议),其会话采用点对点的 resquest/reponse 模式,并且针对移动性有专门的设计。

SIP 提供以下两个基本功能:

1) 会话的建立、调整和终止

SIP 用于发起会话,控制多用户参加的多媒体会话的建立和终结,并能动态调整和修改会话属性,如会话带宽要求、传输媒体类型(语音、视频和数据等)、媒体编解码格式、对组播和单播的支持等。

2) 用户可移动性

在一个 SIP 环境中,用户使用 SIP 统一资源定位器(URL)标识自己。SIP URL 的格式和一个 E-mail 地址相似,通过用户名或主机名等元素来构造,通常由一个 SIP 标识、一个用户名和一个域名组成(例如:SIP:user@company.com)。如果一个用户想被他人找到,则必须向 SIP 服务器登记他当前的位置。SIP 主叫方根据被叫方的公开地址,通过 SIP 重定向/代理服务器查询被叫方当前地址,实现对用户定位,建立会话。

SIP 协议定义了多个实体,分为用户代理和服务器两种。用户代理是呼叫的终端级元素,服务器是处理与多个呼叫相关联信令的网络设备。

用户代理 UA(User Agent)是提供用户交互功能的 SIP 终端实体。用户代理本身又分为用户代理客户端 UAC(User Agent Client)和用户代理服务器 UAS(User Agent Server)。UAC 发起呼叫,UAS 应答呼叫,使点到点的呼叫能够通过客户机/服务器模式完成。

服务器按照功能的不同分为重定向服务器、代理服务器和注册服务器。

重定向服务器通过提供可以选择的位置信息帮助定位 SIP 用户代理,这些位置可能是连通用户的地点。重定向服务器接受经代理服务器转发的用户代理呼叫请求,向代理服务器返回被呼叫用户可能出现的位置列表,并由用户代理直

收稿日期:2004 - 07 - 26;**修订日期**:2004 - 10 - 21

作者简介:杨鹏(1980 -),女,辽宁铁岭人,硕士研究生,主要研究方向:网络与信息安全; 赵博(1980 -),男,陕西西安人,硕士研究生,主要研究方向:网络与信息安全; 王琨(1973 -),男,陕西西安人,博士研究生,主要研究方向:网络与信息安全; 周利华(1942 -),男,江苏人,教授,博士生导师,主要研究方向:分布式多媒体系统、多媒体.

接连接的代理服务器进行用户定位的所有尝试。

代理服务器在会话建立或修订过程中主要起消息的中继转发作用,并且在会话建立时代替呼叫者的用户代理去尝试连接被呼叫用户可能出现的地址,确定被呼叫者的位置。

注册服务器是接受注册的 SIP 服务器,接受本域用户的注册请求,完成用户地址的注册。

这些服务器按逻辑功能的不同进行分类,在实际应用中,可以根据具体情况将若干逻辑实体合并在一个物理实体中,比如某些情况下用户数不多的时候,可以把重定向服务器、代理服务器和注册服务器放在一个物理服务器上实现,并将该服务器称为 SIP 服务器。

2　SIP 实现机制

IETF 提出的 SIP 协议规定了 SIP 系统实现的基本实体、体系结构、工作原理、编码格式、消息结构等,但是对 SIP 呼叫控制系统的实现没有提出一个完整的模型。2001 年 1 月 SUN 公司向广大开发者发布了 JAIN SIP API——一个基于 Java 技术的规范,该规范定义了 SIP 通信实体的模块、接口、工作架构以及流程等。

JAIN SIP 把 SIP 协议规范为标准的 Java 接口,它标准化了协议栈接口、消息接口、事件和事件语意,同时也规定了检测利用其所开发应用的可移植性的方法。用这些接口可实现 SIP 通信中的用户代理和所有服务器。

JAIN SIP 对 SIP 通信实体的实现采用模块化处理,每个实体都由 SipListener、SipProvider 和 SipStack 构成。

SipListener 通常包含在一个应用中,代替应用向 SipProvider 请求 SIP 服务。每个 SIP 实体均是扩展 SipListener 而来,通过在扩展的基础上增加不同的逻辑操作,实现用户代理、代理服务器、注册服务器和重定向服务器等不同的应用。

SipProvider 是 SIP 协议栈向应用提供服务的接口,应用调用的 SIP 服务都由 SipProvider 提供。

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,包括产生地址、消息编解码、消息事件的生成、会话的建立、传输的建立、以及地址的解析等等。SipStack 是 SipListener 和 SipProvider 的基础,其实现的 SIP 服务由 SipProvider 统一向 SipListener 提供。这三部分关系如图 1 所示。

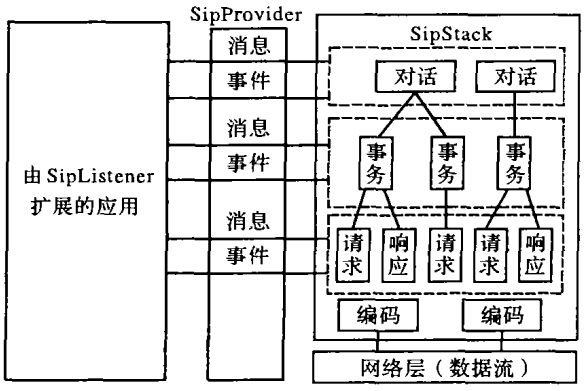

图 1　SipListener、SipProvider 和 SipStack 关系

从图中可以看到应用必然包含 SipListener 或者是由 SipListener 扩展而来,并且由 SipListener 向 SipProvider 申请 SIP 服务。SipProvider 作为 SIP 服务提供接口向 SipListener 提供包括建立对话、产生事务、产生消息体、编解码等全套 SIP 服务,而这些服务具体由 SipStack 执行。

这三个对象的实现也要有规定的顺序。首先通过具体构造函数构建 SipListener,由 SipListener 产生一个 SIP 工厂(SIP Factory),并由 SIP 工厂实例化 SipStack,然后 SipStack 产生 SipProvider,并使 SipListener 与 SipProvider 建立关联。至此,JAIN SIP 各个对象均已建立并产生关联。

同样从图 1 中可以看出,应用中的 SipListener 产生消息发送至 SipProvider,SipProvider 通过与协议栈的监听点绑定来同 SIPStack 建立关联,SIPStack 执行对消息的操作,并且通过网络发送至接收方的 SIPStack。接收方接收到的被称为 SIP 事件,SipProvider 原封不动的将 SIP 事件提交给 SipListener 并由 SipListener 根据事件类型进行具体操作。

3　SIP 通信模型

模拟 SIP 通信的模型具备在局域网内实现点对点呼叫、利用用户名呼叫和跨域呼叫等功能。为了实现这些功能,本模型由用户代理、代理服务器、注册服务器、定位服务器和个性化服务器组成。其中除个性化服务器,其余服务器均与 SIP 规范中定义的 SIP 实体相对应,个性化服务器用来保存域内用户信息和个人设置,如保存用户名和用户 SIP 号的对应关系,使用户只用输入用户名就可以登录,并且可以保存每个用户的地址簿。

3.1　模型结构

图 2 中展示了跨域连接的模型体系结构图。每个域至少有一套代理服务器、注册服务器、定位服务器和个性化服务器。用户通过用户代理登录时,用户代理首先向直接连接的代理服务器发送登录请求,该代理服务器从与它直接连接的注册服务器中查询该用户是否为在本域注册的用户,若是则将该用户当前地址保存到定位服务器并完成登录;若不是则将该登录请求发送至该用户所属域的代理服务器(在实际情况中是该用户购买服务的网络运营商经营的域中的代理服务器),并且将用户当前地址信息存入对应的定位服务器完成用户登录。

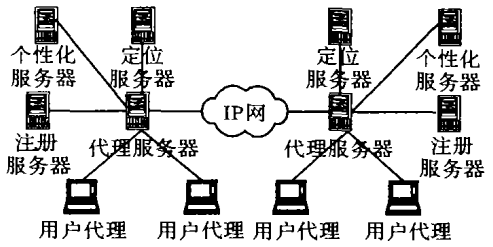

图 2　模型体系结构

当用户 A 向用户 B 发起呼叫时,用户 A 将呼叫请求发送至它直接连接的代理服务器,由此代理服务器将请求转发至用户 B 原始域(实际情况中用户 B 购买 SIP 服务的网络运营商经营的域)的代理服务器,再从该域对应的定位服务器查到用户 B 当前地址信息,并且将呼叫请求转发至该地址,用户 B 对该呼叫请求进行响应,决定是否建立连接。

3.2　模型实现

整个模型用 Java 语言编码,除了个性化服务器之外,其他通信实体均根据 JAIN SIP 架构构建,通过扩展 SipListener 而来,在扩展时添加不同的逻辑操作实现各个实体不同的功

能。本文重点以代理服务器为例重点介绍,其他服务器在利用 JAIN SIP 方面大同小异。

代理服务器是整个系统的核心部分,在整个模型中起重要作用。主要功能为处理接收到的 SIP 请求,向注册服务器转发注册请求;查询定位服务器,获取用户的 sipUrl 地址,根据 SIP 请求向被呼叫方的 sipUrl 进行转发。代理服务器的功能由以下几个模块来完成:

1) 主模块。读取参数初始化代理服务器,启动线程来查询定位服务器中的数据信息并将结果存储,开启 SIP 监听,处理 SIP 的注册、呼叫请求、响应及超时处理。

2) 查询结点信息模块。用来存放 SIP 请求信息,存放于查询队列中。

3) 查询队列模块。用来存放需要进行处理的查询结点信息,分为查询输入队列 incomingQueue 和查询输出队列 outgoingQueue。

4) 查询处理模块。开启线程从查询队列中取出查询结点信息,解析数据后向定位服务器进行查询,并将结果存储回查询队列中。

代理服务器共启动了两个线程,分别为 Proxy 主线程和解决地址对应的 RequestUriResolver 线程。服务启动后,首先初始化 Proxy 服务器,包括产生 SipFactory、SipStack 和 SipProvider,并且设置他们的属性,然后启动 Proxy 线程,即启动消息监听,然后再启动 RequestUriResolver 线程。代理服务器建立 SIP 通信的流程如图 3 所示。

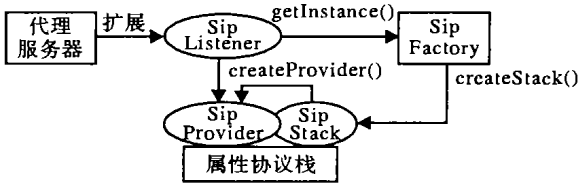

图 3 建立 SIP 通信流程

代理服务器是对 SipListener 的扩展,并且在主线程 Proxy 中先得到 SipFactory 的实例,由此实例生成 SipStack。在设置 SipStack 的各个属性值后,产生 SipProvider,将 SipListener 与 SipProvider 建立关联。这部分代码如下:

```
//得到 SipFactory 的实例
sf = SipFactory.getInstance();
sf.setPathName("net.siptrex");
//由 SipFactory 生成 SipStack
try{
  ss = sf.createSipStack();
}
catch (SipPeerUnavailableException x) {
  log("Error","Siptrex JAIN SIP stack could not be loaded: " +
    x.getMessage());
  System.exit(-1);
}
catch (SipException x) {
  log("Error","Siptrex JAIN SIP stack could not be loaded: " +
    x.getMessage());
  System.exit(-1);
}
//由 SipStack 产生 ListeningPoint
if(listeningPointDefinitionFile != null) {
  try {
    ssi.loadConfigurationFile(listeningPointDefinitionFile);
  }
  catch (SipException x) {
    log("Warning",x.getMessage());
  }
}
//取出 ListeningPoint
Iterator i = ss.getListeningPoints();
if(i == null) {
  log("Error","No ListeningPoints available on the SipStack: " +
    "check configuration file");
  System.exit(-1);
}
//SipStack 产生 SipProvider,并让 SipProvider 与 ListeningPoint 建
//立关联
lp = (ListeningPoint) i.next();
try {
  sp = ss.createSipProvider(lp);
}
catch (ListeningPointUnavailableException x) {
  log("Error","Sip provider could not be created on the " +
    "specified listening point " + x.getMessage());
  System.exit(-1);
}
//SipProvider 与 SipListener 建立关联
try {
  sp.addSipListener(this);
}
catch (TooManyListenersException x) {
  log("Error","Could not add SipListener to the " +
    "SipProvider: Cannot add any more SipListeners to the " +
    "SipProvider" + x.getMessage());
  System.exit(-1);
}
catch (SipListenerAlreadyRegisteredException x) {
  log("Error","Could not add SipListener to the " +
    "SipProvider: Sip Listener Already Registered: " +
    x.getMessage());
  System.exit(-1);
}
```

本模型其余实体(除个性化服务器)的实现在 SIP 通信这部分与代理服务器完全相同,各个实体功能的不同是通过在 SIP 通信的基础上具体逻辑操作的不同实现的,这些操作不属于利用 JAIN SIP 实现通信的范畴,本文不加详述。

4 结语

SIP 是网络通信中的关键技术,IETF 的 RFC3261 和其他一些规范或草案虽然对 SIP 从各个方面进行定义,但是没有从实现级进行更深入的标准化,对工程人员开发应用的通用性和互操作带来了一定的困难。SUN 公司以 Java 语言为基础提出的 JAIN SIP 在一定程度上解决了这个问题,对 SIP 通信的实现提出了模块化架构和一系列标准接口。本文利用这一架构对 SIP 通信实体进行设计,实现了简单的呼叫连接。

参考文献:

[1] CAMARILLO G. SIP 揭密[M]. 白建军, 彭晖, 田敏, 等译. 北京: 人民邮电出版社, 2003.

[2] ROSENBERG J, SCHULZRINNE H, CAMARILLO G, *et al.* SIP: Session Initiation Protocol[S]. RFC3261, IETF, June 2002.

[3] KARAPETKOW S. SIP State of Standards. Siemens Technology and Innovation Development[Z], 2001.

[4] O' DOHERTY P. JAIN SIP Tutorial[EB/OL]. http://java.sun.com/products/jain/, 2004.

[5] HORTON I. Java2 编程指南(SDK 1.4 版)[M]. 李昭智, 王哲明, 王红广, 等译. 北京: 电子工业出版社, 2003.