# Presence 服务中资源列表业务模型的设计与实现①

范绪波[1,2] 雷为民[2] 王 宁[2] 张 伟[2,3] 李俊超[2,3] (1.中国科学院研究生院 北京 100049;
2.中国科学院沈阳计算技术研究所 辽宁 沈阳 110171;
3.中国科学技术大学 计算机科学与技术系 安徽 合肥 230027)

**摘 要:** 随着终端个数的增多，呈现(Presence)服务中的点对点订阅会引起数据量偏大、终端频繁订阅等诸多问题。资源列表服务通过订阅列表的方式很好的解决了这个问题。在参考 SIMPLE 标准的基础上提出一种资源列表业务模型实现框架，详细讨论了列表订阅机制及其设计原理，设计并实现了一种资源列表服务器(RLS)。经过测试，该服务器能正确完成列表订阅，性能良好，很好地解决了终端频繁订阅的问题。

**关键词:** 资源列表; 呈现服务; 资源列表服务器; SIMPLE

## Design and Implementation of Resource List Service Model in Presence Service

FAN Xu-Bo[1,2], LEI Wei-Min[2], WANG Ning[2], ZHANG Wei[2,3], LI Jun-Chao[2,3]

(1.Graduate University of Chinese Academy of Sciences, Beijing 100049, China; 2.Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110171, China; 3.Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)

**Abstract:** With an increasing number of friends, Peer to Peer subscription in Presence causes a great dataflow and many other issues. The Resource List Service is a very good approach to solve this problem by subscribing to the list. This paper proposes a framework for the implementation of Resource List Service Model which refers to SIMPLE standards. Based on this, the RLS as well as its descriptions are designed. Upon testing the RLS in WAN, the experimental results show that RLS can correctly complete the subscriptions of a list and provide an excellent solution for the frequent subscription issues.

**Keywords**: resource list; presence service; resource list server(RLS); SIMPLE

## 1 引言

Presence 即呈现业务，用以传达用户通过一组设备进行通信的能力和意愿，例如：在线、工作中、显示为脱机等。目前 Presence 作为一种引擎业务，可以很方便的为其它业务平台提供业务能力，如即时消息业务平台、一键通、富通信套件(RCS)[1]等。

在 Presence 原有的点对点订阅方法中，用户上线时会向 Presence 服务器订阅他全部好友的 Presence 状态，该用户有多少好友就需要单独发送多少个 SUBSCRIBE[2] 请求，当好友较多时会导致 SUBSCRIBE 信令数量迅速增加，NOTIFY[2]消息随之也会剧增，从而影响服务器性能，在带宽有限环境下，如无线网络中甚至会导致网络拥塞等问题。为了解决上述问题，优化 Presence 服务性能，IETF SIMPLE

① 收稿时间:2010-01-03;收到修改稿时间:2010-02-01

工作组在 RFC4662[3]中提出了资源列表订阅这一概念，这样订阅的时候只需要订阅好友列表对应的 URI，Presence 服务器会返回所有好友的状态信息，不必一个个好友单独订阅，节省了网络带宽资源（这对移动通信比较现实），也易于管理。

本文给出了一种 Presence 服务中资源列表业务模型的实现方案。在 Presence 原有的软硬件平台基础上，添加了一个资源列表服务器(RLS)。文章对比介绍了两种 Presence 业务模型，详细描述了资源列表业务模型设计原理以及资源列表服务器的设计和实现，最后给出了测试结果。

## 2 Presence业务模型

### 2.1 Presence 点对点业务模型

Presence 点对点业务主要涉及三个逻辑实体：信息提供者(Presentity)、Presence 服务器、信息请求者(Watcher)，业务模型如图 1 所示。

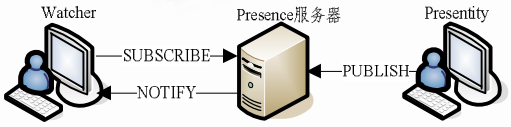


图 1　Presence 点对点业务模型

SIMPLE 标准遵从 RFC 2778[4]和 RFC 2779[5]定义的 Presence 服务模型[6]，同时扩展了 PUBLISH[7]、SUBSCRIBE、NOTIFY 三个 SIP 信令，用以支持这一服务模型。Presence 业务被具体分解为几个步骤：订阅、发布和通知。一个最简单的点对点业务过程如下：一个用户(Watcher) 发送 SUBSCRIBE 请求订阅他感兴趣的其他用户(Presentity)的 Presence 信息，被订阅的用户接收订阅请求，之后每当被订阅用户的信息发生变化，他的终端将会通过 PUBLISH 消息发布自己的新状态到 Presence 服务器，这个新状态随后会通过 NOTIFY 消息通知给订阅者。

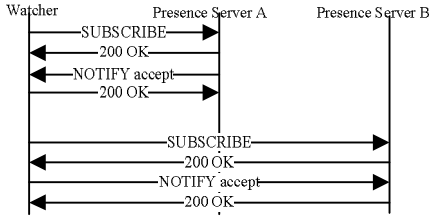


图 2　Presence 点对点订阅信令流程

用户有多少好友就需要发送多少个 SUBSCRIBE 请求到好友所在的 presence 服务器，如图 2 所示，Watcher 有两个好友，分别位于 Presence 服务器 A 和 B，则 Watcher 必须向这两个服务器分别发出好友状态订阅。

### 2.2 Presence 资源列表业务模型

在实现了资源列表服务后，Presence 系统结构如图 3：


图 3　Presence 资源列表业务模型

不管用户有多少好友，终端只需要发一个 SUBSCRIBE 请求到资源列表服务器，订阅自己的资源列表，由 RLS 服务器代替客户端进行全部好友的订阅，RLS 服务器将收到的所有 NOTIFY 消息进行整合，发送一个复合格式的 NOTIFY 消息到终端，完成订阅。如图 4 所示，Watcher 有两个好友，分别位于 Presence 服务器 A 和 B，则 Watcher 只须向 RLS 服务器发出一个资源列表订阅，然后等待 RLS 服务器返回的复合 NOTIFY 消息即可。

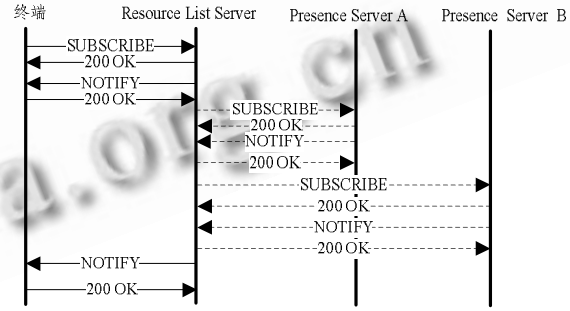


图 4　Presence 资源列表订阅信令流程

## 3 Presence资源列表业务模型的设计原理

为了实现资源列表订阅，首先要能够区分资源列表订阅和点对点订阅，其次要定义一种资源列表的描述方式，并能够对资源列表进行管理配置，最后还要整合多个资源的状态到一个 NOTIFY 消息中。下面就这几个问题对资源列表业务进行详细介绍。

### 3.1 资源列表业务的协商

为了实现资源列表订阅，RFC4662 通过扩展 RFC3625[2]中的 SIP Option tag 机制来支持列表订阅

协商。支持资源列表订阅的客户端需要在 SUBSCRIBE 消息的 Supported 头域中包含“eventlist”，并在 Accept 头字段中指明 multipart/related、application/rlmi+xml 和 pidf+xml 三种格式，以示客户端可以解析这些格式的通知消息；RLS 服务器则需要在 NOTIFY 消息的 Require 头域里包含“eventlist”。资源列表订阅和点对点订阅便区别开来。

### 3.2 资源列表配置管理

资源列表服务中使用基于 XCAP[8]的配置管理方式。XCAP 是一种用以存取用户设置的通信协议，用户可以将特定的配置资料用 XML 的形式存储在服务器上，并允许客户端以 GET、PUT、DELETE 等方法读取、写入、修改、建立和删除数据。

在资源列表业务中，用户首先会通过 XCAP 操作上传自己的 XML 配置文档到 XCAP 服务器，这些配置资料，包括资源列表、好友列表以及授权规则文档等。

### 3.3 资源列表描述文档

RFC4826[9]定义了资源列表文档的格式，并且扩展了两个 XCAP 应用：“resource-lists”、“rls-services”来支持资源列表业务，以便终端可以使用 XCAP 服务器来管理相关配置文档。一是 RLS Services 文档，一是 Resource-lists 文档。前者定义了资源列表服务器上一系列可用服务以及订阅这些服务的 URI；后者是资源列表的具体内容。前者将引用后者的内容。

### 3.4 资源列表状态消息格式

为了构造包含多个资源状态的 NOTIFY 消息体，RFC4662 扩展了 RFC2387 定义的“multipart/related”类型参数，定义了 multipart/related; type="application/rlmi+xml"类型，来描述包含多个资源状态的 NOTIFY 消息体格式。客户端订阅列表时，必须在 SUBSCRIBE 消息中声明对这两种类型的支持。RFC4662 定义的 NOTIFY 消息体分为两部分：RLMI 文档和多个 PIDF 格式的单资源状态文档。

其中的 RLMI 文档描述了该消息体所包含资源的“meta-information”状态，包括授权状态；单资源状态文档描述了列表中具体资源的呈现状态。

## 4 Presence资源列表业务模型的设计与实现

### 4.1 资源列表业务模型实现框架

从图 5 可知这一框架由 RLS 服务器，Presence 服务器，XCAP 服务器，信息提供者(Presentity)、信息请求者(Watcher)五部分组成；其中各个部分作用如下：

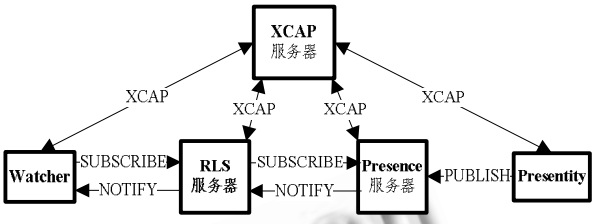


图 5 资源列表业务模型实现框架

(1) 信息请求者(Watcher)：登录时保存自己的 RLS Services 文档和 Resource-lists 文档到 XCAP 服务器，当想要获取好友状态时，发送 SUBSCRIBE 请求到 RLS 服务器订阅自己的资源列表，并对 RLS 服务器返回的整合消息进行解析，获取好友的状态。

(2) RLS 服务器：接收来自终端的列表订阅，通过资源列表文档获取相关资源集合，代替终端发起对资源集合的虚拟订阅，整合所有资源的呈现状态后，返回给信息订阅者。

(3) Presence 服务器：主要处理两种请求，一是信息提供者发布的 PUBLISH，一是 RLS 发送的 SUBSCRIBE 请求。

(4) XCAP 服务器：主要负责保存 RLS Services 文档、Resource-lists 文档以及授权规则文档。

(5) 信息提供者(Presentity)：状态变化时将自己的状态 PUBLISH 到 Presence 服务器，并将授权规则文档上传到 XCAP 服务器，当信息请求者订阅自己时进行相关授权。

### 4.2 资源列表订阅流程

资源列表订阅处理过程需要与 XCAP 服务器配合，由图 6 知，资源列表订阅由以下几个步骤构成：

(1) 终端登录时保存自己的 Resource-lists 文档和资源列表文档到 XCAP 服务器。

(2) 终端向 RLS 订阅他的一个资源列表服务，并且在 SUBSCRIBE 消息的 Support 头域中包含“eventlist”。

(3) RLS 接收到用户的列表订阅请求时向 XCAP 服务器请求该用户对应的资源列表文档，解析该文档并查找 Request URI 所对应的服务。

(4) RLS 对所订阅的服务内容进行关联解析，然后

通过 XCAP 获取该服务所包含的列表资源，对这些资源发起虚拟订阅。

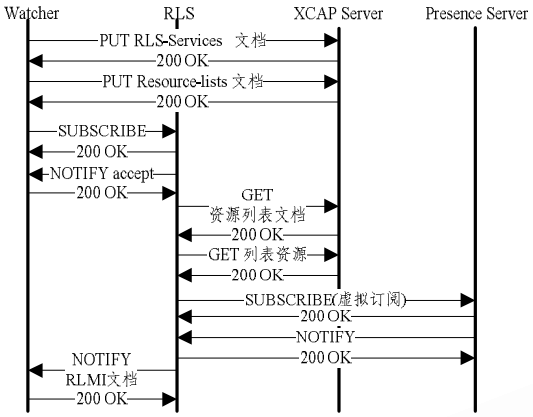


图 6　列表订阅流程

(5) RLS 等待接收虚拟订阅的 NOTIFY 响应，并把所有虚拟订阅的 NOTIFY 响应保存到数据库中，等到所有虚拟订阅响应都收到后对所有状态进行整合，构造携带所有资源状态的 NOTIFY 消息给终端。

(6) 以后每当资源列表中的任何资源状态发生变化时，Presence 服务器都会发送 NOTIFY 到 RLS，RLS 会构造只含有变化资源状态的 NOTIFY 消息到终端，通告某个资源状态发生了变化。

### 4.3 RLS 服务器的设计与实现

#### 4.3.1 RLS 服务器的模块划分

RLS 服务器主要由资源列表订阅处理模块和定时器模块组成，功能模块如图 7，其中各个部分作用如下：

(1) 列表订阅处理模块：采用资源列表的方式，订阅的时候只需要订阅好友列表对应的 URI，一个 SUBSCRIBE 请求会触发多个虚拟订阅，对客户端来讲只需要一个订阅消息就可以订阅所有好友的状态信息。同时处理模块还要对预订资源的 Watcher 列表进行修改。

●资源列表处理模块：完成资源列表相关的操作，主要是完成资源列表的读取和解析，维护相应的原始预订。

●虚拟 Subscription 模块：操作虚拟预订信息，提供其他模块使用的虚拟预订接口。

●Watcher list 信息维护：这个模块根据预订构造 Watcher list 信息，并更改对应 Watcher list 文档信息。

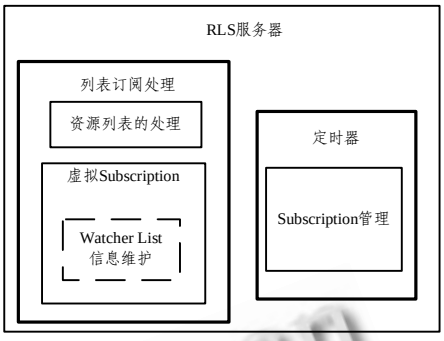


图 7　RLS 服务器功能模块

(2) 定时器模块：这个模块就是为了检测 SUBSCRIBE 消息是否过期，然后调用 Subscription 管理模块进行相关处理。

●Subscription 管理：这个模块负责完成预订的管理，包括预订信息的维护等。

#### 4.3.2 RLS 服务器的工作流程

RLS 服务器的工作处理流程如图 8 所示：

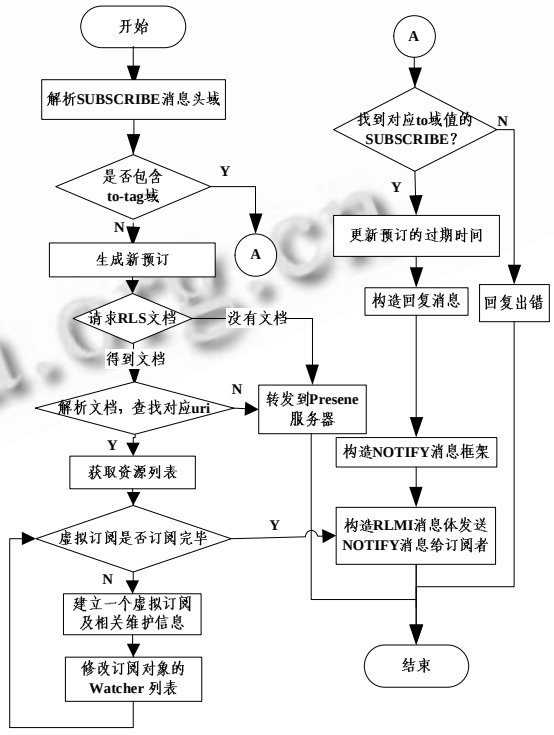


图 8　RLS 服务器工作流程

RLS 收到用户的列表订阅请求时，首先根据 SUBSCRIBE 的 to-tag 头域判断是否为刷新订阅，如果是刷新订阅，则更新订阅信息，构造新的通知消息，

如果不是，则向 XCAP 服务器请求该用户的 RLS Services 文档，解析该文档并查找 Request URI 所对应的服务，通过该服务获取所代表的资源列表，然后对列表中的每一个资源发起虚拟订阅。对于每一个虚拟订阅，都要建立与之相关的维护信息，并修改订阅对象的 Watcher 列表。RLS 等待接收虚拟订阅的 NOTIFY 响应，并把所有虚拟订阅的 NOTIFY 响应保存到数据库，等到所有虚拟订阅响应都收到后对所有状态进行整合，构造携带所有资源状态的 NOTIFY 消息给终端。

笔者一直从事 SIMPLE 标准研究，SIMPLE 标准框架以及 Presence 功能中的点对点订阅已经实现。我们在此基础上实现了 RLS 服务器。

## 5 系统测试

### 5.1 测试平台

测试平台如图 9 所示。测试用例中，用到了三个终端用户 Bob、Alice 和 Eve，分别用到了一台 RLS 服务器、Presence 服务器和 XCAP 服务器。测试情景为用户 Bob 登陆并通过资源列表订阅其好友 Alice 和 Eve 的状态，该平台能够很好的模拟应用场景。

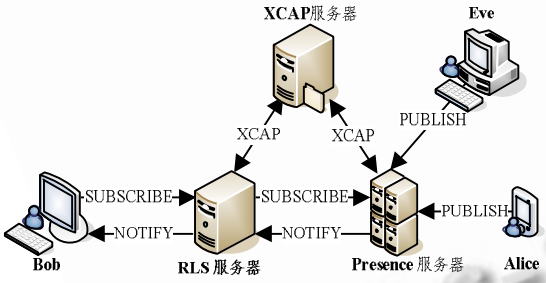


图 9 测试平台结构

### 5.2 测试方法

#### 5.2.1 功能测试

在图 9 所示的测试平台下，通过图 10 所示的流程进行测试，该系统能够成功完成列表订阅，并能够及时通知后续的资源状态变化。可以得出结论：资源列表服务器的功能是成功的。

#### 5.2.2 性能测试

对资源列表服务器进行性能测试，将资源列表中的资源数分别增加到 10、50、100，我们检测网络中的信令数量并与点对点订阅进行对比，测试结果如表 1 所示：

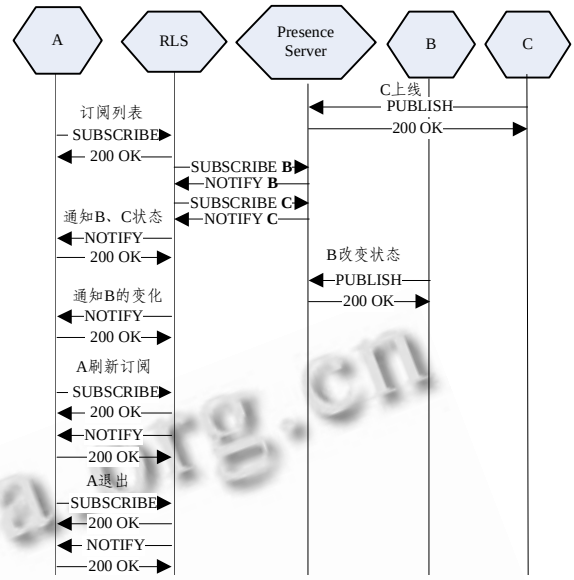


图 10 测试信令流程

表 1 测试信令数据

| 资源数 | 点对点订阅信令数 | 列表订阅信令数 |
|---|---|---|
| 10 | 40 | 6 |
| 50 | 200 | 6 |
| 100 | 400 | 12 |

测试结果显示出了资源列表服务器具有较好的性能，能有效减少网络流量。

## 6 结论

为了避免 Presence 点对点业务的频繁订阅问题，借鉴 SIMPLE 标准，提出了一种资源列表服务实现框架，并在这一框架基础上详细讨论了资源列表相关机制。笔者在RCS服务器研究课题中，以Linux操作系统为平台，实现了支持资源列表服务的原型系统，经实验测试，基本功能已完成并运行良好，能够正确处理用户的好友列表订阅，有效减轻了 Presence 服务器负载。

## 参考文献

1 GSMA. Rich Communication Suite White Paper Version 1.0.October 2008.

2 Roach A. Session Initiation Protocol (SIP)-Specific Event Notification. Internet Engineering Task Force, Request for Comments (Informational) RFC 3265,

June 2002.

3 Roach A, Campbell B, Rosenberg J. A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists. Internet Engineering Task Force, Request for Comments (Informational) RFC 4662, August 2006.

4 Day ML, Rosenberg J, Sugano H, Fujitsu. A Model for Presence and Instant Messaging. Internet Engineering Task Force, Request for Comments (Informational) RFC 2778, February 2000.

5 Day ML, Aggarwal S, Microsoft, Mohr G, Vincent J. Instant Messaging / Presence Protocol Requirements. Internet Engineering Task Force, Request for Comments (Informational) RFC 2779, February 2000.

6 傅之凤,金志刚,李连朋.基于 SIP/SIMPLE 协议实现在席服务. 微处理机, 2004,26(3):19－22.

7 Niemi A, Nokia. Session Initiation Protocol (SIP) Extension for Event State Publication. Internet Engineering Task Force, Request for Comments (Informational) RFC 3903, October 2004.

8 Rosenberg J, Cisco. The Extensible Markup Language (XML) Configuration Access Protocol (XCAP). Internet Engineering Task Force, Request for Comments (Informational) RFC 4825, May 2007.

9 Rosenberg J, Cisco. Extensible Markup Language (XML) Formats for Representing Resource Lists. Internet Engineering Task Force, Request for Comments (Informational) RFC 4826, May 2007.