

文章编号:1001-9081(2009)04-1082-05

Symbian 平台 JXTA 对等网络协议的研究与实现

吴佳忻¹, 彭 舰^{1,2}, 张达平¹

(1. 四川大学 计算机学院, 成都 610064; 2. 马里兰州大学 计算机系, 美国 马里兰州 20742)

(juicewoo@gmail.com)

摘 要:介绍了 Symbian 平台及目前移动平台上已有的资源共享模式,通过分析现有移动 P2P 协议 JXME,提出 Symbian 平台上的新型 P2P 资源共享模式,参考 JXME 进行了一系列改动和调整,包括中继的改进和 Symbian 端协议的重新实现,为 Symbian 平台搭建了相应的 P2P 协议 JXTA-Symbian,最后对 JXTA-Symbian 的实现框架,特别是对 Symbian 手机端协议的实现进行了详细的阐述和分析。

关键词:Symbian 平台;资源共享;P2P;JXME

中图分类号:TP393.04 **文献标志码:**A

Research and implementation of JXTA P2P protocol on Symbian platform

WU Jia-xin¹, PENG Jian², ZHANG Da-ping¹

(1. College of Computer Science, Sichuan University, Chengdu Sichuan 610064, China;

2. Department of Computer Science, Maryland University, Maryland 20742, USA)

Abstract: The Symbian platform and current models of resource sharing on mobile devices were introduced firstly. Next, a new P2P model of resource sharing on Symbian devices was put forward by analyzing the existing mobile P2P protocol JXME. And then the corresponding P2P protocol JXTA-Symbian for Symbian platform was constructed through a series of modifications and adjustments on reference protocol JXME, which included the improvements of relay and the re-implementation of Symbian-End protocol. Finally, this paper discussed and analyzed the implementation architecture of JXTA-Symbian, especially the implementation of Symbian-End protocol, in detail.

Key words: Symbian platform; resource sharing; P2P; JXME

0 引言

众多移动智能设备中,Symbian 移动智能设备以其强大的 CPU 和大容量存储,以及可编程的操作系统脱颖而出,占据了市场的主流。然而和其他移动智能设备一样,Symbian 平台在提供资源共享服务时沿用了较传统的模式:端对端直连方式和以网络或者网络节点为中介的方式。这两种方式虽然较成熟,却有对重要的中间节点依赖性过强和网络规模不可随意扩大等缺陷。为解决这些问题,Sun 公司提出了 JXME 协议^[1]并给出了 Java 语言的实现方案,但该方案在 Symbian 平台上的实施效果却不合人意,执行效率较低,甚至容易出现内存泄漏问题。

因此,本文参考该协议提出并实现了基于 Symbian 平台的 JXTA 对等网络协议——JXTA-Symbian,从而为 Symbian 平台上提供一种新的移动资源共享方式——P2P 模式:Symbian 平台将通过 JXME 中继代理挂载到一个搭建好的 P2P 网络,实现与其他以同样方式连入到 P2P 网络的移动平台之间的资源共享。这种方式改善了集中处理模式的瓶颈,降低了网络负载,同时也提高了端对端资源交换的速率。

1 Symbian 平台和 JXTA-Symbian 简介

Symbian OS 是专为移动设备所设计的实时、多任务的纯 32 位操作系统,是真正的微核操作系统,即操作系统只有很

小一部分运行在最高优先级,其他的功能以 Client-Server 方式提供,具有模块化的系统结构,提供了良好的扩充空间,支持强大的通信及多媒体功能,具有功耗低,内存占用少等特点^[2],是目前主流的,全球市场占有率最大的智能手机操作系统^[3]。

JXTA-Symbian 协议是参考 JXME 协议的实现原理专门为 Symbian 平台构建的。JXME 协议由两部分组成:JXTA 中继代理模块以及 JXME 手机端协议的实现实体^[4]。它为 CLDC 和 MIDP 平台提供 JXTA API,使其能挂载到 JXTA 对等网络中,与 JXTA 网络的其他对等体相互通信。

本文在对 JXME 协议进行分析和实验后,采用 Symbian C++ 实现新的 Symbian 手机端协议,解决了原手机端程序由于访问 Symbian OS 系统资源受限,内存管理不善以及执行效率较低等问题。JXTA-Symbian 协议实现也是由两部分组成:改进后的中继代理模块以及 Symbian 手机端协议实体。协议整体实现及内部事务如图 1 所示。

2 存在问题及分析

在 JXME 中继代理模块实际运行过程中时常出现消息包丢失,数据包发出后连接中断等问题,这些问题具体而微,却严重影响中继性能,为了提供真正可靠稳定的传输,必须要对中继进行一些调整和改进,具体分析参见 3.2 节。

而在手机端协议实现过程中所面对的问题主要源于

收稿日期:2008-10-15;修回日期:2008-12-23。

作者简介:吴佳忻(1983-),女,重庆人,硕士研究生,主要研究方向:分布式系统、移动计算;彭舰(1970-),男,四川成都人,副教授,博士,主要研究方向:分布式系统与中间件、移动计算;张达平(1982-),男,广东广州人,硕士研究生,主要研究方向:分布式系统与中间件、移动计算。

Symbian 平台自身编码规则^[5]的限制以及原有协议的设计两个方面。前者保证内存严格高效管理,同时加深了开发难度,后者通过引入活动对象(Active Object)框架调度同一个线程内的多个异步任务处理,解决协议中涉及到多任务的处理部分造成的内核调度器方面的运行开销。

同时,由于原有协议涉及较多无法避免的静态数据访问,而 SymbianOS 中并不推荐在 DLL 中使用可写静态数据^[6],避免加重内存开销,因此 JXTA-Symbian 中的手机端协议将以服务器框架来实现,并通过客户端接口为第三方应用程序提供服务。这样不仅解决了静态数据的问题,也更便于服务器内部多个活动对象共享数据。

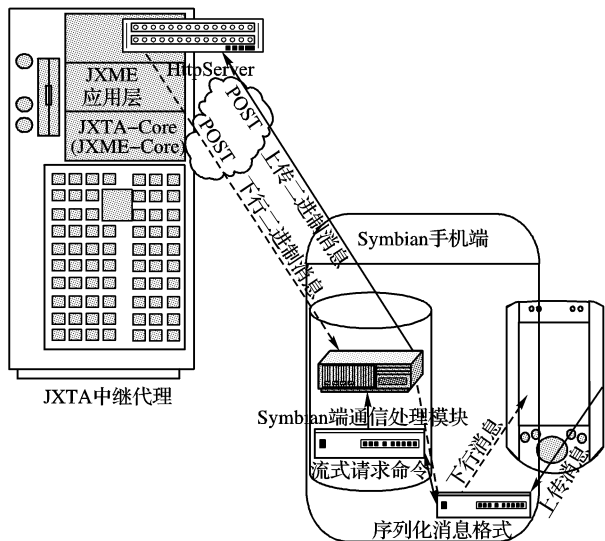


图1 JXTA-Symbian 协议整体实现

3 中继代理端实现原理和改进

3.1 中继代理实现原理

中继代理在外部进程启动后,首先在 JXME-X 应用层模块进行一些必要的资源分配和初始化,并在应用层生成一个监控线程,然后对其他标准 Peer 或者 Symbian 端的消息进行监听及回复。对标准 Peer (包括其他中继) 消息主要依赖实现层中 JXTA-Core 模块 API 的调用,使用发送方的管道发送消息到接受方的输入管道;而 Symbian 端的消息首先要经过 Symbian 手机端协议封装,再启动 HTTP 通信模块发送到中继,然后由 JXME-X 底层模块(与 JXTA-Core 模块同在实现层)将收到的消息加工为 JXME-X 应用层可以分析的不同请求消息,最后 JXME-X 应用层根据消息的类别进行相应的处理。中继整体实现原理如图2所示。

3.2 中继代理参考实现的缺陷以及改进方案

事实上,JXME 中继代理模块运行后,出现一些严重的缺陷,需要对其进行修改和调整,本节详细阐述其中一些主要的缺陷和改进之处^[7]:

1) 中继中消息二级缓存覆盖问题。

在原有实现中,中继代理类把准备发送到手机端的消息存放在一个二级缓存中,需要时再将消息取出,用 `HttpMessageSender.doSend()` 方法将消息发送到中继所在的服务器线程,但实际情况是从二级缓存中取出的消息不等待前面的消息发送出去就继续,容易造成后一个消息覆盖前一个尚未成功发送的消息,所以,应该将 `doSend()` 方法设为同步方法并且在方法内部设置消息的传递控制,等待前一个消息发送成功后再发送下一个消息。

2) 中继和手机端通信时无法支持大数据量上传。

在项目测试时发现,当 Symbian 端点上传的消息在 HTTP 层上形成的数据包大于 1 kB 时,Symbian 端就再也无法接受到正确的返回响应,而且出现 TCP 报错“远程服务器强制关闭连接”。实际上这个问题只需要调整一个 HTTP 服务器参数 `MaxRequestReadTime`,本文依照消息大小将此参数设为 60 000 ms,这个参数决定了服务器的 linger 处理时间,如果数据量过大,在这个时间间隔内服务器不会将处于 Socket 缓冲的数据发送,而直接强制关闭前一连接,便会出现大数据量时发生的“远程连接错误”的 TCP 错误。

3) 中继转发请求消息给手机端时处理逻辑错误。

在 JXME-X 应用层控制中,代理会保存一个影射对(LinkedList 对象),其中存储自身的输入管道和相应的手机端 ID 列表。当代理通过通信模块无法成功发送消息到手机端时,会删除相应 ID。在原有实现中,在删除 ID 后没有更新 LinkedList 迭代子,而本文在删除 ID 后会迭代子及时更新,避免抛出异常。

4) 中继资源发布逻辑改进。

在 JXME-X 应用层的参考实现中,中继在接受创建资源请求后,只发布了一个本地资源广告。在这样的模式下,网络上的其他对等体或者需要很长时间的搜索资源,或者根本无法获得资源广告。在实际运用中应该使用远程发布方式来进行广告发布,以减少其他对等体的资源搜索时间。

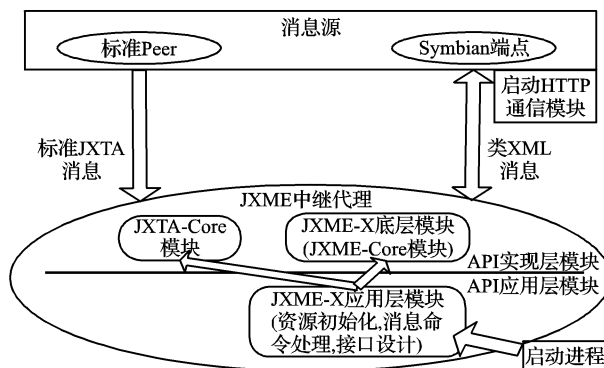


图2 中继代理实现原理

4 Symbian 手机端的设计和实现

为了高效管理客户对系统资源和服务的共享访问,也为了保护系统的完整性和封装静态数据,本文将以服务器框架形式来实现基于 Symbian 平台的手机端协议,客户程序将通过服务器发布的客户端接口来获取所需服务,下面将详细描述服务器内部模块的功能和实现。

4.1 服务器内部模块概述

Symbian 手机端协议的服务器框架内部层次如图3所示。

为了能有效快捷地和中继进行通信,服务器内部需要完成三方面的工作:

1) 统一 Symbian 端点接受/发送的消息格式。消息格式都封装在 `CElement` 和 `CMessage` 类中,形成消息处理层,以便将 Symbian 端点需要处理的数据格式统一为一种“类 XML”消息格式,对于 Symbian 端应用来说,只需要在编程时使用这些固定格式。

2) 统一 Symbian 端点所有的 P2P 命令格式。P2P 命令处理层利用消息处理层将各种的 JXME P2P 命令封装好(包括加入消息头标识),然后通过消息队列依次由通信对象传递到中继代理,同时获取回复消息。

3) 实现 Symbian 端点的通信。主要在消息通信层实现, 使用 Symbian API 提供的 HTTP 协议 GET 方法与中继代理相连, 然后使用 HTTP 协议的 POST 方法上传/下载符合 JXME P2P 命令格式的二进制流化消息。

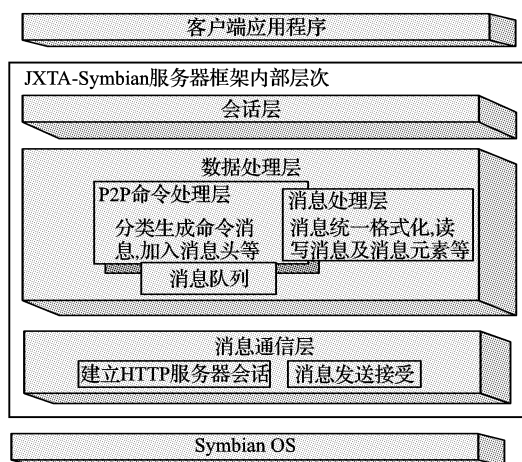


图3 服务器框架层次

4.2 Symbian 端点的消息格式

在 JXME 协议中, MIDP 端点的消息格式被统一为一种“类 XML”的格式^[8]。在本文实现中, 为了让 JXta 中继代理能够正确识别消息格式, Symbian 端的“类 XML 格式”参照原有格式分为两个表示部分: 消息 CMessage 类以及消息元 CElement 类。每个 CMessage 对象由 CElement 元素组成, CElement 对象是存储数据的基本单元。而 CMessage 对象由具体的 CElement 对象组成, 只是在组成的 Element 之前导入了统一的 Message 头而已。这里的 JXME 消息实际上, 指的是 CMessage 格式的对象。Symbian 端点通过通信层, 获取到二进制消息后, 需要通过一个反序列化的过程 (CMessage.ReadL()), 将流式的二进制消息转化为格式良好的 CMessage 对象。而 Symbian 端点要发送消息的时候, 会通过一个相反的过程, 将 CMessage 格式的消息序列化为二进制流化消息 (CMessage.WriteL()), 进而通过通信层发送到 JXta 中继代理。序列化与反序列化的过程就是将 CMessage 对象以及 CMessage 对象之内的 CElement 对象逐字节写入/读出的过程。

在此过程中 CMessage 以及 CElement 的对象格式必须是相对固定的, 这样才能被 Symbian 端点和 JXta 中继代理共同识别。根据 JXME 协议规定, CMessage 对象以及 CElement 对象的二进制流化消息将形成特定的内部格式。

表1 CMessage 对象内部格式

| Jxmg (标志) | 0 (版本) | 命名空间 个数 | 命名空间 列表 | Element 个数 | Element 列表 |
|--------------|-----------|------------|------------|---------------|---------------|
|--------------|-----------|------------|------------|---------------|---------------|

表2 命名空间列表格式

| | | | | |
|----------------|----|-----|----------------|----|
| 单个命名空 间字符长度 | 名称 | ... | 单个命名空 间字符长度 | 名称 |
|----------------|----|-----|----------------|----|

表3 CElement 对象内部格式

| Jxel 格式 | 命名空 间序号 | 0/1 | 名称 长度 | 名称 | 类型 长度 | 类型 | 数据 长度 | 数据 |
|------------|------------|-----|----------|----|----------|----|----------|----|
|------------|------------|-----|----------|----|----------|----|----------|----|

4.3 Symbian 端点的 P2P 命令格式

利用客户应用程序所传递的数据和消息处理层, 可以较容易地形成 JXME-Symbian 端的 P2P 命令。只需要将应用程序发出的 P2P 所有命令按照消息格式模块的定义, 转化为相应的命令消息 (包括 Join 命令 (加入 P2P 对等组) 请求消息,

Search 命令 (搜索资源) 请求消息, Create 命令 (创建资源) 请求消息, 对 pipe 操作 (监听管道, 关闭管道以及发送消息到管道) 的请求消息), 然后在这些消息头部加上便于 JXta 中继代理进行识别 P2P 的统一发送标志, 最后由通信层将其传递到 JXta 中继代理。

主要命令分别转化成表 4~7 的消息格式:

表4 Join 命令 (加入 P2P 对等组) 请求消息

| Element 名称 | Element 数据 | Element 命名空间 | Element |
|------------|----------------------|--------------|---------|
| Request | Join | Proxy | Null |
| Id | NewGroupID, HBufC8 * | Proxy | Null |
| Arg | Password, HBufC8 * | Proxy | Null |
| Request Id | 消息 request ID | Proxy | Null |

表5 Search 命令 (搜索资源) 请求消息

| Element 名称 | Element 数据 | Element 命名空间 | Element |
|------------|---------------------|--------------|---------|
| Request | Search | Proxy | Null |
| Type | Attr, HBufC8 * | Proxy | Null |
| Value | query, HBufC8 * | Proxy | Null |
| Thre-shold | Threshold, HBufC8 * | Proxy | Null |
| Request Id | 消息 request ID | Proxy | Null |

表6 Create 命令 (创建资源) 请求消息

| Element 名称 | Element 数据 | Element 命名空间 | Element |
|------------|----------------|--------------|---------|
| Request | Create | Proxy | Null |
| Name | Name, HBufC8 * | Proxy | Null |
| Id | Id, HBufC8 * | Proxy | Null |
| Type | Type, HBufC8 * | Proxy | Null |
| Request Id | 消息 request ID | Proxy | Null |

表7 pipe 操作请求消息

| Element 名称 | Element 数据 | Element 命名空间 | Element |
|------------|-------------------|--------------|---------|
| Request | Listen/close/send | Proxy | Null |
| Name | Name, HBufC8 * | Proxy | Null |
| Id | Id, HBufC8 * | Proxy | Null |
| Type | Type, HBufC8 * | Proxy | Null |
| Request Id | 消息 request ID | Proxy | Null |

最后, 在这些消息的头部加上 P2P 的统一发送标志, 以便 JXta 中继代理进行识别。一般说来, 公共的消息头格式如表 8。

表8 公共消息头格式

| Element 名称 | Element 数据 | Element 命名空间 | Element |
|-------------------------------|--|--------------|---------|
| EndpointDestiation Address | 代理地址/EndpointService: jxta-NetGroup/urn: jxta: uuid- 32 位码 | Jxta | Null |
| EndpointSource Address | Jxta: // 自身的 peerId | Jxta | Null |

加上消息头的发送消息, 被存储在 Symbian 端的一个发送缓存队列中。当 Symbian 端点启动与 JXta 中继代理的会话过程后 (启动一个独立轮询线程, 以一个固定的时间间隔发送请求消息, 接受代理的回复消息), 存储在发送缓存中的请求消息就会按放入的顺序被逐一传递到 JXta 代理端。

4.4 Symbian 端通信模式

服务器内的消息通信层主要由类 CHttpMessenger 实现, 主要功能是为 Symbian 端点提供 JXta 的消息通信服务, 在

Symbian 端点与 JXTA 中继代理之间初次成功建立关系后,实现接受与读取二进制消息。整体流程如图4所示。

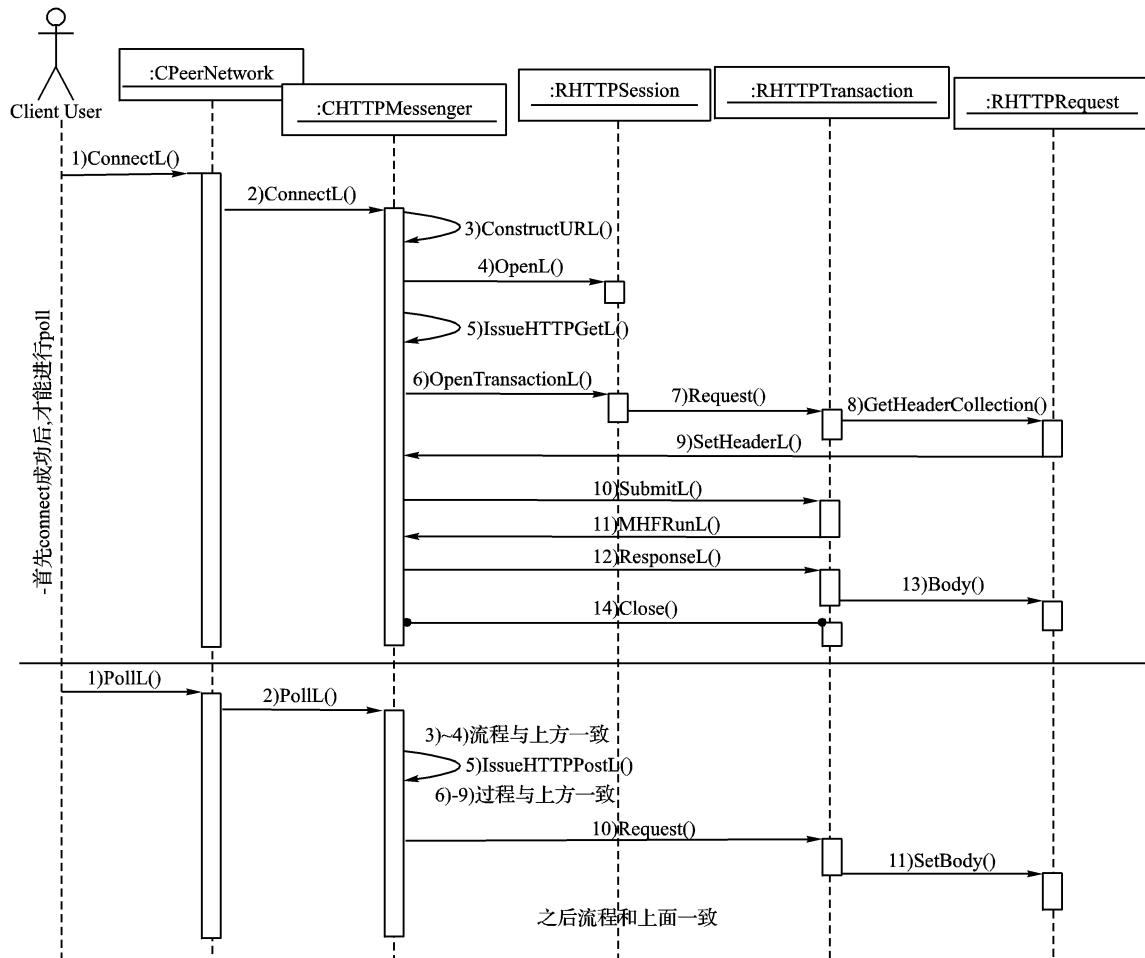


图4 通信模块整体流程

4.4.1 客户端程序和中介建立连接

首先,在客户端程序和中介能够互传消息之前,必须和中介成功地建立连接。即必须成功调用图(4)中的 ConnectL()。其流程如下:

1)客户端程序通过客户端接口请求建立连接,首先服务器调用 CPeerNetwork ::ConnectL(),发布连接命令。

2)通过 CHTTPMessenger ::ConnectL()将命令传递给通信模块。

3)利用命令中包含的原始 URL,通过 CHTTPMessenger ::ConstructL()构造出连接中继的 URL。这个 URL 具有以下格式:

[HTTP://address:port]/pid? Relay poll Time[连接有效时间], lazytime out [JXTA peer-1], RelayURL [HTTP://address:port]/uuid-DEADBEEFDEAFBABAFEEDBABE000000F05 [Endpoint serviceID]/EndpointService: jxta-NetGroup/Command[具体的指令]。

4)通过 Symbian OS API 提供的 RHTTPSession ::OpenL()建立一个 Symbian OS HTTP 服务器会话。

5)~10) CHTTPMessenger ::IssueHTTPGetL()用 Symbian OS API 下的 RHTTPTransaction GET 方法向中继发起一个已建会话下的事务,并通过 RHTTPRequest 设置 Request 头。

11)~13)由于 RHTTPTransaction 在其自身内部实现时使用了活动对象,因此会返回事件信号,通过 CHTTP ::MHFRUNL()可以获取事件信号,并作出相应处理,例如通过 RHTTPTransaction ::Response()取得响应消息。

14) CHTTPMessenger ::Close()关闭连接,释放占用资源。

4.4.2 客户端发送消息并轮询等待回复

在成功连接后,客户端程序可以发送消息并开始轮询中继的回复消息。流程如下:

1)客户端程序通过客户端接口请求轮询,首先服务器调用 CPeerNetwork ::PollL(),发布轮询命令。在轮询过程中,会将消息队列中封装好的消息发送到 CHTTPMessenger 对象中。

2)通过 CHTTPMessenger ::PollL()将消息传递到通信模块。

3)~4)和 4.4.1 中的 3)~4)过程基本相同。

5)~9)和 4.4.1 中的 5)~9)这段过程基本相同,只是 CHTTPMessenger ::IssueHTTPPostL()使用 Symbian OS API 下的 RHTTPTransaction POST 方法。

10)~11)通过 RHTTPTransaction 和 RHTTPRequest ::SetBody()为使用 POST 方法的通信对象设置将被 POST 的 Request 消息体。

12)之后的步骤和 4.4.1 中的 11)~14)基本相同,即等待并获取中继响应消息。

5 JXME 和 JXTA-Symbian 对比

完整的 JXME 协议由 JXME 手机端协议和 JXME 中继代理模块共同组成,用 Java 实现整个协议,代码小巧易用,为 CLDC/MIDP 平台提供 JXTA API,支持发现管道、组和对等体,创建管道和对等组,加入对等组等基本功能,从而使 MIDP 平台可以通过 JXTA 网络与外界对等通信。

JXTA-Symbian 协议是由 Symbian 手机端协议和改进后的 JXME 中继代理组成,为 Symbian 平台提供 JXTA API,本节将对两者异同作出仔细的比较,并由此说明 JXTA-Symbian 协议的必要性。

在平台支持上,JXME 支持 CLDC/MIDP 平台,由于 Symbian OS 7.0 提供了 MIDP2.0 实现,因此 JXME 也支持部分 Symbian 平台,而 JXTA-Symbian 专门针对 Symbian 平台,包括 S40,S60,S80,UIQ 等系列。

在开发语言上,JXME 选用 Java 语言实现,虽然 Symbian OS 支持 Java 语言,但由于 Java 语言是解释型,它与将源码编译成本地代码并直接执行的编译型 Symbian OS C++ 语言不同,在执行 Java 编译出来的 class 文件之前,Symbian OS 需要进行诸如启动 Java 虚拟机、解压所需的 jars 和进行 JIT 等一系列操作。相对于 Symbian OS C++,繁琐且低效;而 JXTA-Symbian 手机端协议采用 Symbian OS C++ 来实现,作为 Symbian 原生语言,C++ 在内存使用和执行效率具备最佳性能。

在系统资源的操控能力上,由于 Symbian OS 对系统资源的保护,并未将操作系统资源的 API 全面提供给 MIDP 平台,因此,JXME 参考实现对 Symbian 平台系统资源的操控能力较弱,例如不能直接对文件进行随机访问;而 Symbian OS C++ 为 JXTA-Symbian 手机端实现提供了最全面的 Symbian OS APIs^[9],不言而喻,对系统资源的操控更加容易。

在多任务处理问题上,JXME 利用了 Java 的多线程机制,包含了一些数据共享方法,然而在 Symbian OS 中启动一个线程至少需要 16kb 堆栈空间,而且线程间频繁的上下文切换消耗更多系统资源,运行速度低;JXTA-Symbian 手机端协议实现则引入活动对象框架^[10]来处理多任务,所有活动对象都在同一个线程内调度,占用空间小,共享数据方便,活动对象之间上下文切换的开销远远低于线程间切换,资源消耗小,运行效率高。

在内存管理上,JXME 依托于 JVM 和 Java 的垃圾回收机制来管理内存,而 Symbian 平台要求更严格的管理机制,采用了两阶段构造分配内存,清理内存则更加复杂,使用了清理栈,异常处理等机制,以防止内存泄漏,JXTA-Symbian 手机端实现在内存管理上也是这样要求的。

通过各方面比较,可以看出,虽然 JXME 协议面向的领域和平台更加广泛,但在 Symbian 平台上却并不完全适合,不仅执行效率低,也容易造成数据传输错误,系统内存泄漏等严重问题。而 JXTA-Symbian 协议不仅实现了 JXME 中的基本功

能,而且针对 Symbian 平台的具体特性,在 Symbian 端协议实现上采用了适当的机制,高效安全地与 Symbian OS 结合在一起,使 Symbian 平台能够真正地加入 JXTA 对等网络。

6 结语

本文参考 JXME 协议提出了基于 Symbian 平台的新的 P2P 网络协议——JXTA-Symbian,在对 JXME 实现原理进行详细分析后,改进了 JXME 参考实现中中继代理的主要缺陷,并针对 Symbian 平台独有的特性,采用 Symbian OS C++ 重新实现了 JXTA-Symbian 的手机端协议。最后,通过对比 JXME 和 JXTA-Symbian 的异同,证明了 JXTA-Symbian 的有效性和必要性。

参考文献:

- [1] JXTA Official Community. Welcome to the JXTA for J2ME (JXME) project [EB/OL]. [2008-06-29]. <https://jxta-jxme.dev.java.net>.
- [2] SALES J. Symbian OS internals: Real-time kernel programming [M]. Hoboken, NJ: John Wiley & Sons, 2005: 1-16.
- [3] Symbian Ltd. Symbian fast facts [EB/OL]. [2008-06-10]. <http://www.symbian.com/about/fastfacts/fastfacts.html>.
- [4] KHAN F. Wireless messaging with JXTA, Part1: Using JXTA technology [EB/OL]. [2008-06-08]. <http://www.ibm.com/developerworks/wireless/library/wi-jxta/>.
- [5] BABIN S. Developing software for Symbian OS: An introduction to creating smartphone application in C++ [M]. Hoboken, NJ: John Wiley & Sons, 2005: 81-112.
- [6] STICHBURY J. Symbian OS explained: Effective C++ programming for smartphone [M]. Hoboken, NJ: John Wiley & Sons, 2004: 218-223.
- [7] 丁磊,李志蜀,彭舰. 深入解析 JXME 的 MIDP 端协议以及视频共享框架 [J]. 四川大学学报: 自然科学版, 2007, 33(4): 2-3.
- [8] SIDDQUI B. JXTA4J2ME Implementation Architecture [EB/OL]. [2008-06-11]. <http://www.developer.com/java/j2me/article.php/1501461>.
- [9] Symbian Developer Official Community. Programming languages in Symbian OS guide [EB/OL]. [2008-06-11]. http://www.symbian.com/Developer/techlib/v70sdocs/doc_source/DevGuides/ProgLanguages.html.
- [10] SHETTY G. Symbian active object framework [EB/OL]. [2008-06-11]. <http://www.newlc.com/Symbian-Active-Object-Framework-A.html>.

(上接第 1081 页)

4 结语

Parlay 与 SIP 之间的映射包括两个方向,本文的映射模型设计了 Protocol_Decode 与 SCF_Responder, Protocol_Decode 与 SCF_Responder 这两对类来分别两个方向消息映射的处理。在本文所搭建的测试环境中,通过实现一个完整的 SIP 请求和响应过程,证明了本文所设计模型的正确性和可行性。由于本文在设计中采用了基于 CORBA 的方式,因而使得模型更为简单灵活,增强了模型的独立性。当然,本文所设计的映射功能较为简单,若要使 Parlay 网关支持一个较为 Parlay API 第三方业务,则需要对此模型做进一步的扩展,以实现 Parlay 网关的其他各项功能。

参考文献:

- [1] 万晓榆,姚平香,樊自甫. 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.

- [2] 韩艳峰,卢美莲. Parlay API 与 SIP 间呼叫相关概念映射的研究 [J]. 北京邮电大学学报, 2005, 4(2): 59-62.
- [3] CHUNG K S, CHOI Y I. An interworking mechanism between SCFs and protocols in the open service gateway [C]// Proceedings of the 2006 Asia-Pacific Conference on Communications: APCC'06. Busan: [s. n.], 2006: 1-4.
- [4] 罗仕漳,武家春,廖建新. Parlay 网关 SIP 协议映射关键技术研究 [J]. 计算机应用研究, 2006, 23(12): 125-126.
- [5] OMG. CORBA/TC Interworking and SCCP Inter-ORB protocol specification [EB/OL]. (2004-06-12) [2008-08-15]. <http://www.omg.org/technology/documents/formal/corba.tc.interworking.and.sccpi.htm>.
- [6] OMG. CORBA 语言映射 [M]. 韦乐平,薛君敖,孟洛明,译. 北京: 电子工业出版社, 2001.