基于 EPCglobal 初级阅读器协议的简单模型的设计与实现①

Design and Realization of Simple Model Based on EPCglobal Low-Level Reader Protocol

蔡　磊　(复旦大学 计算机科学技术学院 上海 200433)

摘　要： 在 RFID 大量应用的今天，RFID 阅读器和客户端之间的通信协议并没有完全统一。为此，EPCglobal 组织 2007 年推出了初级阅读器协议 Low-level reader protocol(LLRP)。在对其进行了详细的分析和研究的基础之上，以两台 PC 机为平台，设计了一个模拟客户端、RFID 阅读器间消息发送和响应过程的简单模型来实现初级阅读器协议 LLRP。通过模型的设计和最终实现，表明了 LLRP 作为客户端和 RFID 阅读器端接口的可行性和有效性，为最终实现整个 EPCglobal 系统打下了坚实的基础。

关键词： 初级阅读器协议 EPCglobal 射频识别 C# TCP/IP Xml

1　引言

RFID 射频识别是一种非接触式的自动识别技术，它通过射频信号自动识别目标对象并获取相关数据。在传统的 RFID 应用和开发中，虽然 RFID 阅读器最终是实现一样的功能，但是当 RFID 阅读器在硬件或软件上稍微有任何变动，那么控制阅读器的客户端的软件也必须随之改变。当用户采用不同厂家提供的产品时，其整体的维护成本也随之提高，这造成开发工作的重复性，系统之间的不兼容性和维护的复杂性。

在阅读器协议方面，2007 年 4 月，RFID 标准化组织 EPCglobal 推出了针对阅读器和客户端之间的初级阅读器协议(low-level reader protocol，以下简称 LLRP)[1]。LLRP 是一个用来描述从客户端到阅读器的协议，如图 1 所示。

LLRP 的出现改变了支持同一功能的无数的私有协议接口存在的局面，使得软件开发者能更专注于客户的需求。

一些开源组织已经针对 LLRP 使用了 Perl、Java 开发了相应的工具包，包括了用来描述 LLRP 不同命令及阅读器和客户端间协议的软件代码库。但只是从协议上实现了软件接口，并未从客户端、LLRP 和阅读器端整个系统架构的层次来完整地实现 LLRP。本文试图建立一套简化的 LLRP 规范的实施模型，并采用.NET 的 C#语言进行开发。为尝试采用 LLRP 规范的开发人员提供一个参考。

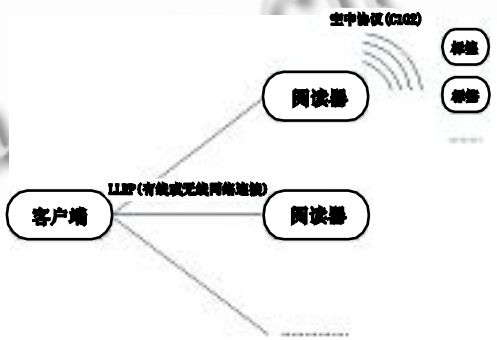

图 1　处于阅读器和客户端之间的 LLRP

2　LLRP标准分析

2.1 LLRP 的基本数据定义

LLRP 基本的数据类型包括消息、参数和域，LLRP 是以消息为基本数据单元来通信的。通过消息，客户

① 收稿时间:2009-02-08

端能控制各种各样的动态数据结构(例如：ROSpecs、AccessSpec 等)和并以此来配置和操作阅读器的行为。阅读器也通过消息的方式来回应自己的状态信息或返回客户端所需要的标签数据和其它响应信息。其中消息可以包含一个或多个参数和域，一个参数也可以包含一个或多个域。

消息包含：

(1) 指出此消息所属协议版本号的版本值。

(2) 能唯一标识此消息的消息类型值。

(3) 消息 ID:由于在阅读器内部处理消息的时间长短的问题，所以阅读器对消息的响应并不能和客户端发送消息的顺序一致。消息 ID 用来将阅读器响应的消息和客户端发送的消息相连接。

(4) 此外，它还包含了强制的和可选的参数。

参数用来在 LLRP 消息中传送 LLRP 操作的特殊细节，每个参数包括：

(1) 在一个消息内能唯一标识此参数的参数类型值。

(2) 此外，它还可以包含个体域和子参数。

域：消息和参数可以包含个体域，域一般以枚举的方式出现。

一个典型的 LLRP 消息的数据结构如图 2 所示：

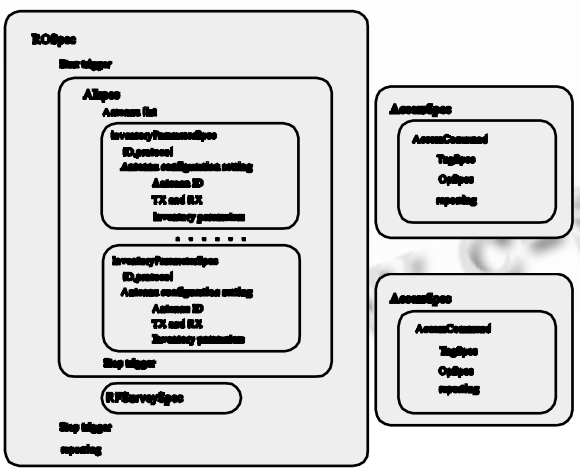

图 2　一个典型的消息的结构

在 LLRP 规范中主要有两个被定义的数据结构,所有这些定义只是基本的数据定义，在实际使用时可以根据需要灵活变动:

(1) 一个是 ROSpec(阅读器操作规格)：ROSpec 最重要的方面是它包含着一个或多个 AISpecs(它定义了怎样从天线来读取 EPC 标签)和一个 RFSurvey(它定义了怎样得到像射频功率这样的天线操作信息)。此外，ROSpec 还包含了一个 ReportSpec(它有一个描述一个报告包含什么信息的参数)。

(2) 另一个主要的数据结构是 AccessSpec，它定义了怎样读标签上的非 EPC 信息(例如用户定义数据) 和怎样在标签上做其它的操作，例如写和灭活。

2.2 RFID 阅读器和客户端消息的传输过程

LLRP 位于阅读器和客户端之间，空中协议位于阅读器和标签之间，LLRP 和空中协议虽然是属于不同的范畴，但是通过与阅读器之间的互动，客户端的软件获取其在标签上的自己所需信息，一个典型的处于客户端、阅读器和标签之间的 LLRP 和空中协议互动过程如图 3 所示：

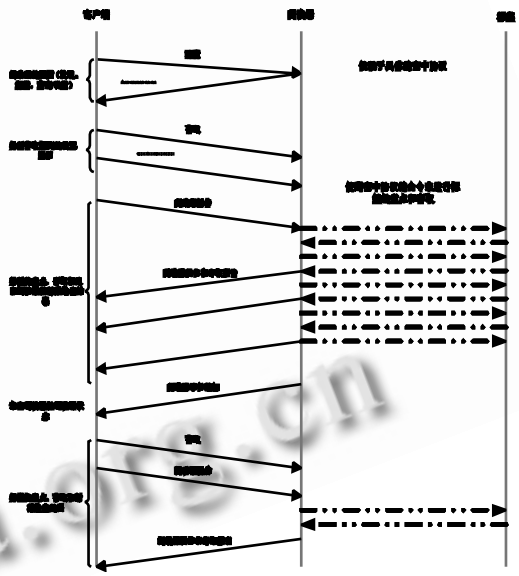

图 3　LLRP 和空中协议的互动

本文更关注 LLRP 的实现过程。整个 LLRP 的操作包括下面这么几个阶段：

(1) 首先是阅读器的配置，包括阅读器的能力发现、配置和查询设置，阅读器的能力发现包括阅读器的天线数、一般目的输入(GPI)端口数、支持的射频检查等等。客户端发送这样的消息过去之后，阅读器发送自己的信息和配置查询后的响应消息回客户端。

(2) 然后是标签存取规则的设置、更新。这取决于客户端应用时所采取的具体空中协议。

(3) 在前面的连接配置工作做好以后，这里开始真正的阅读器操作。阅读器操作包括天线和标签的盘

点、标签的存取和对阅读器射频的检查。

(4) 此时再进行第二次阅读器操作的循环，直到停止事件触发为止。

3 LLRP消息的实现和简单模型的架构设计

LLRP 详细规格参数是通过 LLRP 消息来实现的，所以要实现 LLRP 规范，首先应先定义好所有的 LLRP 消息。

3.1 LLRP 消息两种不同的实现方式

由上面 LLRP 的分析，可以看出 LLRP 是基于消息的。LLRP 消息内又嵌套地包含着参数和域。这种数据结构的定义在具体实现时对编程语言的有效性和效率都构成了挑战。而在现实中，LLRP 被组织成两种不同的模式：面向对象和面向文档。

(1) 面向文档的消息实现方式是基于 XML 和 XPath 的。它没有对象的层次关系，消息结构明了易读。只使用 llrp.xsd，程序设计者就可以很容易的构建一条面向文档的消息。

(2) 在面向对象模式中，具体的消息和面向对象中预先定义好的对象一一对应，对象树依据“创建和附加”的编程方式创建起来。通过建立好的对象树，程序设计者可以依次在程序中设计自己需要的消息对象。面向对象的消息实现方式执行效率更高。但是创建一个嵌套了很多层参数和域的消息可能会包含很多行代码，既不好创建也不利于阅读。

3.2 面向文档的 LLRP 消息的实现

由于 LLRP 消息的嵌套性，即消息可以包含一条或多条参数和域。结合 XML 标签的可嵌套与 XML 的可扩展性，以及使用 XML 通信可解决数据格式和通信协议不一致的优点，XML 用来作为描述 LLRP 消息的二进制格式是再合适不过了[2]。例如将 LLRP 中的 ADD_ROSPEC 消息表示为如下 XML 格式：

由图 4 所示，标签 ADD_ROSPEC 在 LLRP 中为消息，标签 ROSpec、ROSpecStartTrigger、ROSpecStopTrigger、ROBoundarySpec、AISpec、AISpecStopTrigger、InventoryParameterSpecID 在 LLRP 中为参数，Priority、ROSpecID、CurrentState、ROSpecStartTriggerType、ROSpecStop TriggerType、DurationTriggerValue、AntennaIDs、AISpecStopTriggerType、DurationTrigger、InventoryParameterSpecID、ProtocolID 为域。

```
<ADD_ROSPEC Version="1" MessageID="1">
 <ROSpec>
  <ROSpecID>1</ROSpecID>
  <Priority>0</Priority>
  <CurrentState>Disabled</CurrentState>
  <ROBoundarySpec>
   <ROSpecStartTrigger>
    <ROSpecStartTriggerType>Immediate</ROSpecStartTriggerType>
   </ROSpecStartTrigger>
   <ROSpecStopTrigger>
    <ROSpecStopTriggerType>Duration</ROSpecStopTriggerType>
    <DurationTriggerValue>0</DurationTriggerValue>
   </ROSpecStopTrigger>
  </ROBoundarySpec>
  <AISpec>
   <AntennaID>1 2 3 4</AntennaID>
   <AISpecStopTrigger>
    <AISpecStopTriggerType>Null</AISpecStopTriggerType>
    <DurationTrigger>0</DurationTrigger>
   </AISpecStopTrigger>
   <InventoryParameterSpec>
    <InventoryParameterSpecID>1</InventoryParameterSpecID>
    <ProtocolID>EPCGlobalClass1Gen2</ProtocolID>
   </InventoryParameterSpec>
  </AISpec>
 </ROSpec>
</ADD_ROSPEC>
```

图 4　消息 ADD_ROSPEC 的 XML 描述

3.3 面向对象的 LLRP 消息的实现

在实现面向对象的 LLRP 消息时，将消息和参数都采用对象的方法来进行存取，每个具体的消息都是一个类，每个具体的参数也是一个类，域按照 LLRP 规范作为消息类和参数类的成员变量。如果在 LLRP 定义中某个参数包含在某个消息内，那么这个参数就作为这个消息的成员变量，并在这个消息的构造函数中实例化这个参数，使他们同时生成。这个消息类和它所包含的参数类是一种合成关系。由于 LLRP 规范中消息数量众多，这里以消息 ADD_ROSPEC 为例来显示 LLRP 消息类的 UML 描述，如图 5 所示。

其它的 LLRP 消息也可同样根据 LLRP 规范写成对应的类，由于 LLRP 提供可扩展性，对于类的定义可以通过在对应的消息类和参数类中加扩展的参数类或者成员变量的方式来达成，对于对应的 XML 文档，通过添加对应的标签也可符合要求。

3.4 简单模型的架构设计

在现实的程序设计中，程序员可能会采用这两种方式，所以本文在模型设计中采用了这两种方式，以便获得更好的兼容性。整个模型的架构如图 6 所示。

在模型中，客户端和阅读器端通过 TCP/IP 通信，基于 XML 文档的消息作为 TCP/IP 两端传输和接受的数据流[3]。在两端各自收到对方的消息后，将文档化

的消息转化为已经定义好的消息对象，并将文档消息中的参数一一复制给对象类对应的参数，对象通过参数的改变与其它消息对象实现互动，或触发其它消息状态的转换，或直接执行对标签的读取，进而最终实现整个 RFID 和客户端之间的互动[4]。

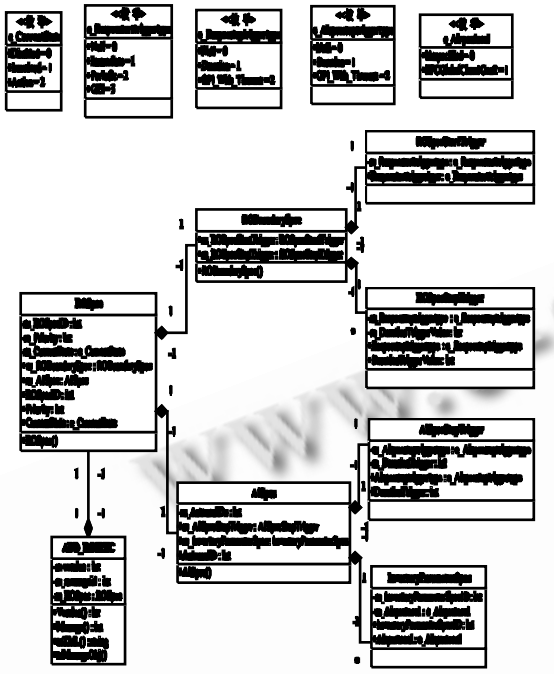

图 5　ADD_ROSPEC 类的 UML 描述

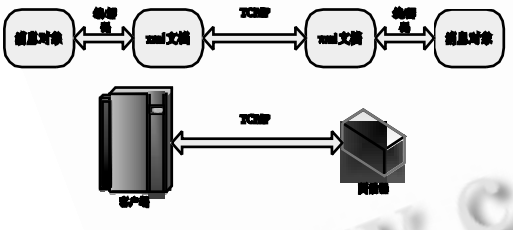

图 6　模型架构图

(2) 阅读器端到客户端的消息响应：阅读器端通过对客户端发送来的 ADD_ROSPEC 消息对象值的判断，产生对应参数值的响应消息 ADD_ROSPEC_RESPONSE，然后将其转换成对应的 XML 流，再通过 TCP/IP 发送到客户端去，客户端收到阅读器端发送的 XML 流后将其转换成对应的消息对象以供客户端使用。

```
IPAddress address = IPAddress.Parse("192.168.1.61");
TcpListener serverListener = new TcpListener(address, 11002);
try
 {serverListener.Start();
  TcpClient tcpClient = serverListener.AcceptTcpClient();
  NetworkStream netStream = tcpClient.GetStream();
  ADD_ROSPEC r = new ADD_ROSPEC();
  r. toMessageObj(netStream);
  ..............
  sr.Close();
    netStream.Close();
    tcpClient.Close(); }
catch (Exception re)
  {MessageBox.Show(re.Message);}
finally
  {serverListener.Stop();}
```

图 7　客户端部分程序

4　LLRP模型的实现和实验结果

4.1 LLRP 模型的实现

本地消息对象和 XML 流之间的转换(编/解码)是通过在消息类中的成员函数来实现的，如图 5 中消息 ADD_ROSPEC 的 toXML()和 toMessage()两个成员函数就是用来进行编解码的[5]。

为了显示 LLRP 在客户端和阅读器端的实现，依据模型架构图实现客户端到阅读器端的消息发送过程和阅读器端到客户端的消息响应这两个过程：

(1) 客户端到阅读器端的消息发送过程：通过客户端生成一个消息 ADD_ROSPEC 的实例对象，将其转换成对应的 xml 流，然后通过 TCP/IP 将 XML 流发送到客户端，阅读器端接受到客户端发送来的 XML 流后，通过判断 XML 流中 MessageID 标签的值来将其内容赋值给对应的消息对象。

阅读器端在接受到客户端发送来的消息后，根据对应的消息类型和变量值会做出相应的动作，并对阅读器端做出响应，发出相对应的消息对象到客户端，这个过程和上面客户端到阅读器端的的过程类似，也同样使用 C#提供的那些类来实现它。这里就不一一叙述了。

4.2 实验结果

在客户端发送命令到阅读器端被触发、响应到客户端收到响应结果这一整个模型实现过程中，两个分别模拟阅读器端和客户端的 PC 上已经按 LLRP 定义

(下转第 151 页)

(上接第 137 页)

的对象值和 XML 消息值的改变符合预期，这一实验结果表明了 LLRP 能在此简单模型下得以基本的实现,简单模型的合理性和有效性得以验证。由于模型的设计和实现基于 LLRP 的典型消息，要完整地实现 LLRP,必定要引入更多的消息和触发过程，此简单模型的可靠性和稳定性将在以后的深入研究中得到更进一步的改进。

5 结论

通过对初级阅读器协议(low-level reader protocol)中消息的数据定义的引入和在客户端和阅读器之间的传输过程详细地介绍了 LLRP 规范。在此基础之上，结合实际应用的需要，设计了针对 LLRP 规范的简单模型，并通过模拟客户端和阅读器端之间消息发送和响应，实现了基本的 LLRP 过程，为后续全面实现 LLRP 的奠定良好的基础。实验结果表明采用 LLRP 规范能有效的实现客户端和 RFID 阅读器之间的信息交换功能，从而印证了 LLRP 的可行性和有效性。这将为最终实现整个 EPCglobal 打下了坚实的基础。

参考文献

1 EPCglobal Inc. Low Level Reader Protocol(LLRP). V1.0.1,2007-08-13,http://www.ecpglobalinc.org/standards/llrp.

2 孙更新,裴红义,杨金龙.XML 完全开发指南.北京:科学出版社, 2008.

3 史蒂文斯.TCP/IP 详解卷 1:协议.北京:机械工业出版社, 2000.

4 潘晓君.RFID 中间件数据传输的研究.中国西部科技, 2007,(02).

5 Troelsen A. Pro C# with .NET 3.0. Special Edition, Berkeley:Apress, 2007.