RFID READER AGENT BASED ON LOW LEVEL READER PROTOCOL (LLRP) STANDARD

Sidhartha Samanta
MS in Physics

Project Report Committee
Dr. Craig Thompson (Chair)
Dr. Amy Apon
Dr. Gordon Beavers
Agenda

- Introduction
  - Problem
  - Objective
  - Approach
  - Potential Impact

- Background
  - Middleware
  - RFID
    - EPCglobal and LLRP

- Architecture
- Implementation
- Conclusions
  - Summary
Introduction

- **Problem**

- Lack of communication standards between RFID readers and RFID networks has prevented widespread usage of RFID technology and limits interoperability between competing RFID technologies.
Introduction

Objective

- **TagCentric** is an open source agent-based application capable of integrating RFID readers, RFID tag printers, and a database of the user’s choosing. Developed at University of Arkansas.

- **Low Level Reader Protocol** (LLRP) is a specification published by EPCglobal (a standards-setting body) in April 2007 that is aimed at providing a way for enterprises to communicate with readers using a standard API.

- The **objective** of this project is to extend TagCentric by developing and integrating an LLRP reader agent (**LLRPReaderAgent**) that can talk to any RFID reader conforming to LLRP standards.
Introduction

Approach

- Joe Hoag under Dr. Thompson’s supervision developed a Java based LLRP class library in June 2007, now available on SourceForge.

- The TagCentric middleware project comprises several kinds of RFID reader agents covering different manufacturers of RFID readers, e.g. SymbolReaderAgent.

- The TagCentric reader client (LLRPReaderAgent) developed for this project provides a way for TagCentric’s XML reader messages to map to and control a LLRP-enabled reader.
Introduction

- LLRPReaderAgent uses the messages defined in LLRP library to communicate with the reader.

- The LLRPReaderAgent serves as a TagCentric wrapper for an LLRP-compliant reader.

- LLRPReaderAgent communicates with the TagCentric application via well-defined XML messages, and communicates with an LLRP reader via LLRP standard byte-stream messages.

- For testing the client, a virtual LLRP reader developed by a company called Pramari is used.
Introduction

Potential Impact

- A successful implementation of this project will enable the TagCentric RFID middleware application to communicate with any reader that is LLRP compliant.

- Since TagCentric, the LLRP class library and the LLRPReaderAgent are all open source software, this work can be used by the open source community to further develop RFID software solutions, including solutions compatible to the LLRP EPCglobal standard specification.
Background

- **Middleware**

  - Middleware is a software layer that resides between application and operating system layers.

  - It facilitates building large enterprise systems by connecting small application systems.

  - Middleware hides complexity such as location and access transparency, transaction management, marshalling and unmarshalling of arguments for remote procedure calls, asynchronous messaging, fault tolerance and priority scheme that developers need not worry about while building complex and enterprise wide business applications.
The following are examples of kinds of middleware:

- Message Oriented Middleware
- Object Request Brokers (ORB)
- Remote Procedure Calls
- Transaction Processing Monitors
Radio Frequency Identification (RFID)

- RFID is an automatic identification technology.
- It relies on cheap tags (transponders) attached to objects such as vehicles, pets, pallets, and containers which a reader (antenna and a transceiver) in the nearby vicinity can read (measured in inches to yards).
- The main difference between RFID and bar code technology is that RFID eliminates the need for line-of-sight reading that bar coding depends on.
RFID systems are used in retail-industry where it is being used to track pallets and containers in big warehouses, for solving out-of-stock problem and to reduce inventory.

Some auto manufacturers use RFID systems in their assembly lines. At each successive stage of production, the RFID tag tells the computers what the next step of automated assembly is.

RFID systems can be used just about anywhere, from clothing tags to tanks to pet tags to food -- anywhere that a unique identification system is needed.
EPCglobal and LLRP standards

- EPCglobal is an organization that develops standards for RFID.

- Recently, EPCglobal published a document that specifies a standard interface between RFID readers and enterprise clients, the Low Level Reader Protocol (LLRP).

- The interface provides a means for the clients to retrieve reader capabilities, control reader operations, and facilitates robust error handling and status reporting. LLRP protocol communication data units are called messages which can be from reader to the client and vice versa.
Background

- Messages from the client to the reader include getting and setting configuration of readers, capabilities discovery of readers and managing the inventory and access operations at the readers.

- Messages from the reader to the client include the reporting of reader status, RF survey, and inventory and access results.

- This application layer, message-oriented protocol not only standardizes the way clients and readers communicate with each other, it also allows for air protocol specific operation and provides a mechanism for reader vendors to define vendor specific extensions.
High Level Design

Starting in the Spring of 2005, the “Everything is Alive” pervasive computing project in the Computer Science and Computer Engineering Department of the University of Arkansas developed the Ubiquity Agent System and, using Ubiquity as a base, the TagCentric RFID Middleware application.

The Ubiquity Agent architecture assumes that agents of various kinds or classes (e.g., devices, databases, GUIs) can communicate with each other using XML messages sent over a bus-like architecture.
Architecture

Figure 1: Sample TagCentric Configuration
The LLRP standard, published in April 2007 by EPCglobal, specifies an interface between RFID readers and clients. The interface specification includes the following:

- Commands to control RFID reader operations such as to inventory, read, write and kill tags.
- Robust status and error reporting.
- Ability to discover reader capabilities.
- Support for vendor specific extensions.
In June 2007, an LLRP class library was developed by Joe Hoag, a Ph.D student under Dr. Thompson.

LLRP is a *bytestream-based* protocol composed of many messages and each of those messages can have several parameters associated with them.

All LLRP message classes inherit from an abstract base class `Message` shown in Figure 2.
Figure 2: Message Base Class
The Message class contains methods to serialize message objects into bit streams and methods to deserialize bit streams into corresponding message objects.

The base Message class includes two methods to serialize a message object namely, serialize() and send().

The send() method first calls the serialize() method and then sends the serialized output to an OutputStream.
The receive() method is the reverse of the send() method. It deserializes the bit streams from InputStream into the corresponding message object.

The getMessageType() can be used to identify the type of the incoming message.

The toXMLString() and toXML() methods can be used to show the output in XML format. The show() method prints the XML output to standard output.
Architecture

- All LLRP parameter classes inherit from an abstract base class Parameter as shown in Figure 3.

- There are two general types of parameters in LLRP namely Type-Value (TV) and Type-Length-Value (TLV). The classification is based on the way they are serialized and deserialized.

- All TV parameters inherit from TVParameter base class and all TLV parameters inherit from TLVParameter base class.
Figure 3: Parameter Base Classes
To serialize a Parameter object into a byte stream, the serialize() method can be used. The deserialize() method can be invoked to convert the bit streams into a parameter object.

The getParameterType() method can be called to identify the type of parameter object.
LLRPReaderAgent (my project)

- LLRPReaderAgent extends the base ReaderAgent class.

- It has a `processReaderOn()` method that enables the gathering of RFID tag data from the LLRP reader.

- It also implements a method called `processReaderOff()` method that disables the gathering of RFID tag information from the Reader.

- LLRPReaderAgent also has a method called `processTerminate()` that kills all local threads, shuts down the socket and finally kills the Agent base class.
Figure 4: Reader Agents in TagCentric
Implementation

Implementation Context

- SourceForge LLRP class library implementation efforts are ongoing in Java, C++ and Perl.

- Since the TagCentric is implemented in Java and this project is building on the LLRP Java implementation, the LLRPReaderAgent is also implemented in Java.
Implementation

- **LLRP Messages implemented by LLRPReaderAgent**

  - There are several messages that a client can execute to communicate with a LLRP compliant reader.

  - The following sections describe the activities and messages that the TagCentric LLRPReaderAgent performs to instantiate, turn on, turn off and terminate a Reader. TagCentric only exercises a subset of the LLRP reader operations.
Implementation

- **Instantiation**

  - In the instantiation phase, the LLRPReaderAgent performs the following activities:
    - Creates a **socket** to communicate with the LLRP Reader.
    - Launches a **thread** that acts as a listener to capture any reply message from the LLRP Reader.
    - Sends the **ADD_ROSPEC** command to the Reader.
    - Sends the **ENABLE_ROSPEC** command to the Reader.
A Reader Operation (RO) specifies the operations to be executed at one or more antennas.

A RoSpec describes the operational parameters for a RO.

The Reader Client issues the ADD_ROSPEC command (setting various parameters to specify variables such as Antenna Id, Protocol Id, priority, trigger, and parameters such as duration) which put the ROSpec in disabled state.

The subsequent message issued by the Client namely the ENABLE_ROSPEC transitions the state to inactive.
Implementation

Figure 5: LLRP Messages and Reader Action in Instantiation Phase
Implementation

- Enable Reporting ("ReaderOn")

- In this phase, the LLRPReaderAgent sends the START_ROSPEC message which takes the ROSpec to an active state and enables the Reader to emit tags in the form of RO_ACCESS_REPORT messages.

- The listener thread that was launched in the instantiation phase captures the RO_ACCESS_REPORT message from the Reader and converts it to a generic tag format applicable for all the reader types supported by TagCentric application.
Figure 6: LLRP Messages and Reader Action in ReaderOn Phase
Implementation

- **Disable Reporting ("ReaderOff")**

  - In this phase, the LLRPReaderAgent sends the `STOP_ROSPEC` message which puts the ROSpec in an inactive state and the Reader stops emitting tags.
Implementation

Figure 7: LLRP Messages and Reader Action in ReaderOff Phase
Implementation

- Terminate

- In the terminate phase, the LLRPReaderAgent performs the following activities:
  - Sends the `DISABLE_ROSPEC` command to the Reader.
  - Sends the `DELETE_ROSPEC` command to the Reader.
  - Kills the listener thread activated in the instantiation phase.
  - Closes the socket created in the instantiation phase.
  - The sequence of events in the terminate phase is the reverse order of events executed in the initialization phase.
Figure 8: LLRP Messages and Reader Action in Terminate Phase
Result

- The new TagCentric LLRPReaderAgent was tested using a virtual LLRP reader developed by Pramari, a Connecticut based company that develops different kinds of RFID virtual readers to simulate real readers in research labs.

- The following slides describe the steps we followed to test the new LLRPReaderAgent from the TagCentric application.
Result

- **Launch Pramari Virtual Reader**
  - Download and install the Pramari Virtual Reader available in Sourceforge.
  - Create a new Reader of type LLRP in the Pramari Virtual Reader GUI.
  - Generate and add tags to the antenna of the reader created in the previous step.
  - Started the LLRP reader.
Result

Figure 9: Pramari Virtual Reader
Result

Create new LLRP Reader Client in TagCentric

- Created a new Reader Client of type LLRP in TagCentric admin panel specifying the reader name, type, IP address as shown in the screen print below.
Figure 10: Creating a new LLRP Reader Client in TagCentric
Figure 11: New LLRP Reader Client in TagCentric
Result

- Instantiate the LLRP Reader

  - The LLRP reader client was instantiated by pressing the launch button against the new client created in previous step.

  - As described in the implementation section of this report, the steps in this phase such as creating the socket and listener thread, sending the RO spec messages and responses from the virtual reader were executed successfully.
Result

- **Enable the LLRP Reader**

  - The LLRP Reader was enabled by switching to the reader panel in TagCentric GUI and pressing the Enable button.

  - The reader status changed to enable as expected. The test panel was then selected, allowing for verification that the tags from the Virtual Reader appear on the screen.
Result

Figure 12: Enabling the LLRP Reader Client in TagCentric
Result

Figure 13: Tags as read from LLRP Virtual Reader
Result

- Disable the LLRP Reader

  - The LLRP reader was disabled by pressing the Disable button on the reader panel. The tag generation stopped as expected.
Result

- **Terminate the LLRP Reader**

  - The LLRP reader was successfully terminated by pressing the kill button associated with the reader on the main Admin panel.

  - The activities in this phase such as RO Spec messages, destruction of the listener threads and closing of the socket were executed successfully.
Figure 14: Terminate LLRP Virtual Reader
Limitations of the current test

- No actual LLRP reader yet exists in the market.
- The Pramari virtual LLRP reader is still very much a prototype and it needs restarting after each reader cycle comprising of launch, start, stop and terminate phases.
Conclusion

Summary

- This project enhanced the open source TagCentric RFID middleware application by developing and integrating into it a new reader agent LLRPOReaderAgent capable of communicating with any reader that conforms to LLRP standards.
Conclusion

- **Future Work**

  - Testing was done against a virtual reader in its infant stage.

  - One of the future goals in relation to improving the LLRPReaderAgent is to test it against a fully implemented virtual reader by Pramari and also against an actual LLRP reader.
The current version of the LLRPReaderAgent only implements a small subset of messages that the LLRP specification defines.

So, the LLRPReaderAgent can be expanded to include more functionality such as air protocol commands to implement singulation (identifying an individual tag in a muti-tag environment), RF survey operations (an operation during which the reader performs a scan and measures the power levels across a set of frequencies at an antenna), discovering reader device capabilities, and complex tag operations such as locking tag memory and killing tags.