Smart Device Virtualization: 
Building an LLRP Reader Emulation Tool

Kyle Neumeier

Thesis Director: Dr. Craig Thompson
Thesis Committee: Dr. Amy Apon, Dr. Dale Thomspn
Problem

- Businesses need better data
  - To understand operational processes
  - For asset management
- High overhead associated with RFID
  - Hardware
  - Software
  - Training
  - Implementation
- Reasons that RFID is expensive:
  - RFID implementations are not "off the shelf"
  - Current processes might need to be modified
  - Lack of standards
Objective

- Design and Implement a virtual Low Level Reader Protocol (LLRP) RFID reader.
Approach

- **Standards**
  - Low Level Reader Protocol

- **Virtualization & Simulation**
  - Pre-purchase evaluation
  - Repeatable development
  - Incremental integration
  - Training

- **Rifidi RFID simulation tools**
  - Open source developed by Pramari LLC
  - Aims to lower cost of entry to RFID solutions
  - Allows users to experiment with virtual reader processes and data

- **Advantages of Rifidi**
  - Virtual reader framework
  - Other Rifidi applications can take advantage of it
Background - RFID

- **Radio Frequency Identification (RFID)**
  - Used for a method of automatic identification
  - Has roots in Identify, Friend or Foe (IFF) transponders in WW II
  - Applications: Passports, Toll Collection, Animal Identification, Healthcare

- **EPCglobal**
  - Standards body for RFID technology
  - Born from Auto-ID Center at MIT
  - Specifications
    - Class 1 Gen 2 (C1G2)
    - Application Level Events (ALE)
    - EPC Information Services (EPCIS)
Background - LLRP

- Aims to standardize reader-client interface
- Uses data units called messages
  - Command messages sent from client to reader
  - Response and event messages sent from reader to client
- Client Configures reader using data structures (specs)
- Makes use of triggers - Events that can start and stop reads
Background - LLRP

- LLRP Messages are bit-encoded

- Reader Operation Spec (ROSpec)
  - Configures how Inventories are done

- Access Operation Spec (AccessSpec)
  - Configures how data operations are performed on a tag
Background - LLRP

- **Triggers**
  - Events that cause transitions in RO Specs and Access Specs

- **Start Triggers**
  - Timer
  - GPI Events

- **Sop Triggers**
  - Timer
  - GPI Events
  - Tag Count
Background - LLRP

- State diagram of ROSpec
  - Messages are in all capital letter
  - Trigger events are in lower case
Background - LLRP
Background – Rifidi Product Suite

- **Rifidi Emulator**
  - Aims to be an Integrated Development Environment for RFID Applications Developers
  - Contains framework needed to build RFID emulators

- **Rifidi Tag Streamer**
  - Load Testing tool for edge servers.
  - Allows user to define high-volume tag batches and scenarios

- **Rifidi Designer**
  - Business process simulation tool
  - Allows user to drag and drop components in to a 3D environment and define and run scenarios.
Architecture – Rifidi Emulator

- Written in Java
  - Portability
  - Large API
  - Allows integration with eclipse
- Divided into two main components
  - User Interface
  - Engine
Architecture – Rifidi Emulator

- **User Interface**
  - Eclipse-based
    - SWT
    - Plugin Architecture (OSGI)
    - Common plugins already exist (text editors, etc)
    - Library of UI Components (wizards, views, etc)

- Allows users to create readers and tags
- Users can drag and drop tags on to reader’s field of view
Architecture – Rifidi Emulator

- **Engine**
  - Framework for building RFID reader emulators
  - Provides commonly needed components
    - Messaging framework (TCP/IP, Serial, etc)
    - Message processor
    - Radio (For reading tags)
    - RFID Tags
    - General Purpose I/O
Architecture – Rifidi Emulator

- Engine and UI communicate via RMI
Architecture – Rifidi Emulator
Architecture – Rifidi Emulator

- Communication
  - Responsible for maintaining connections and sending and receiving messages between client and reader
  - Abstraction layers handle various kinds of physical connections, such as TCP/IP, UDP, and serial
  - Mostly generic for all readers
Architecture – Rifidi Emulator

- **Message Processor**
  - Handles message parsing and execution of command
  - Each virtual reader implements a message parser that to parse messages in a specific way
  - Parsed commands are sent from the message processor to command handler methods
Architecture – Rifidi Emulator

- **Radio**
  - Component that handles tag operations
  - Contains
    - Antenna – External interface for adding and removing tags
    - Tag Memory – Reader-specific structure for keeping track of tags. Every reader implements its own Tag Memory
    - Radio – Interface between antenna and Tag Memory
Architecture – Rifidi Emulator

- General Purpose I/O
- GPIO is serial interface for communication of events with other devices
  - Electronic eye, sensors, push arms, etc.
- Rifidi Engine provides support for GPI/O events via GPIOController
Architecture – LLRP Reader

- LLRP Reader Extends Rifidi architecture
  - LLRP Messages
  - Connection
  - Reader Operation Controller
  - Triggers
  - Access Operations
Architecture – LLRP Reader

- LLRP Messages
- Uses LLRP Toolkit (developed at U of A) to translate between LLRP messages and Java objects
Architecture – LLRP Reader

- **Connection**
- **LLRP Specification** requires that a LLRP reader be able to both accept and initiate LLRP Connections
Architecture – LLRP Reader

- Reader Operation Controller
  - Manages state of ROSpecs.
  - Handles Execution of ROSpecExecutor Threads
- Triggers
  - Uses Observer/Observable design pattern
Architecture – LLRP Reader

- Access Operation Controllers
  - Handles state of Access Operations
- Required tag classes that modeled C1G2 tags
- Allows virtual reader to write, lock, kill, and read data from tags
Implementation

- **Iteration 1: Basic Command Support**
  - Added just enough functionality to get tag reads back
  - The reader could process basic ROSpecs

- **Iteration 2: Improved Triggering and Event Notifications**
  - All start and stop triggers supported
  - Event notification messages

- **Iteration 3: Access Command Support**
  - Involved redesign of tag classes
  - AccessSpecController that moved access specs between states

- **Iteration 4: General Purpose I/O**
  - GPIO Controller written; GPIO perspective added to IDE
  - RMI callback functionality needed to support monitoring of GPO

- **Iteration 5 and 6: Suspend and Unit Test**
  - Suspend and resume support future pause and play of Rifidi Designer
  - Unit tests model EPCglobal’s conformance test of LLRP
Conclusions – Lessons Learned

- Other, non-RFID virtual devices could be written using a similar message processing paradigm.
- Some might be able to make use of Rifidi components and architecture.
- Other devices would be useful in higher level simulations.
Conclusions – Future Work

- Allow virtual LLRP reader to support future vendor extensions
- Build other kinds of virtual smart devices that can interact with Rifidi
- Construct other clients that can talk to the RMI interface
- Use a XML Rifidi client interface with Second Life simulations
- Integrate LLRP support into Tag Centric
Conclusions - Successes

- Rifidi is a “best in show” finalist at the RFID Journal Live trade show in Las Vegas
- Virtual LLRP reader will be used to test IBM’s Premesis RFID edge server
- Penn State will use LLRP as a training platform for upcoming classes
- Contributions to LLRP toolkit