RFIDentify: Speaker Presence Sensing

Blake Arnold, Willi Ballenthin
{baa2121,wrb2102}@columbia.edu

Internet Real-Time Lab, Columbia University
New York, NY 10027

Internet Real-Time Lab
Spring 2010

Computer Science
Columbia University
New York, New York 10027
URL: http://www.cs.columbia.edu/irt/

RFIDentify: Speaker Presence Sensing

Blake Arnold, Willi Ballenthin
{baa2121,wrb2102}@columbia.edu

Internet Real-Time Lab, Columbia University
New York, NY 10027

Columbia University
Spring 2010

Abstract

Designing a location aware system to allow large conferences to work seamlessly has been a problem in the past. There have been past try's at trying to create a location aware network and many successful implementations. But none of these successes has yet to allow an instantaneous notification of presence.

1 Introduction

Location aware hardware has been slowly gaining ground over the past few years. The hardware has ranged from Sonar to GPS to cellphone triangulation while prices have decreased. Recently, social networks have become a major motivation in the consumer sector for location aware technology, driving innovation with respect to ease of use. New services have been created to track users of such devices as means of providing relevant recommendations and alerts.

This technology can be used effectively during a large conference with numerous speakers. Keeping track of which speakers are currently presenting is a daunting task for an attendee. It usually requires constant attention to simultaneous schedules and vigilance when tracking updates to the order.

In this paper, we demonstrate a location aware hardware system providing easy and effective speaker identification and notification targetted at large conferences. The main goal of *RFIDentify* is to alert interested attendees when new speakers begin their presentations, while remaining unobtrusive to the presenter.

This system remains unique with respect to previously deployed approaches. Particularly, RFIDentify is implemented with an eye towards event oriented frameworks. It is our hope that RFIDentify might act as a prime example of varied technologies integrated into platforms such as SECE (Sense Everything, Control Everything).

2 Related Work

1. **F8 Conference** - The f8 Developer Conference was held by Facebook in San Francisco on Wednesday April 21. Founder and CEO, Mark Zuckerberg, announced changes for platform developers. Provided to each attendee of the event was a token containing an RFID chip. By tapping the token at various points throughout the venue, attendees could share their activities, tag themselves in photos, and connect to other attendees. RFIDentify looks to provide similar functionality, with particular emphasis of open and independent distribution of event notifications.

2. **Marc Petit-Huguenin** - Marc Petit-Huguenin has described the inefficiencies of IETF meetings with regard to presenter identification, and proposes solutions involving RFID readers. He is particularly concerned with the privacy of such systems, and has gone to lengths to design custom RFID readers and card holders. The RFIDentify project aims to incorporate his considerations and hardware reader into its goals.

3 Outline

In Section 4, we introduce the use cases of the new software and hardware combination. In Section 5 describes are strategy and design of the new system with further details in Section 6. Finally, we describe future work in Section 7.

4 Problem

Following IETF presentations remotely requires a lot of work by the end user. They must download the slides in a variety of formats (such as PDF, DOCX, or PPT), determine who is speaking by an XMPP message and then, perhaps, listen to the audio feed through a video streaming program. The system has a heavy reliance on data entered by humans, such as the current speaking, uploading slides and speaking into the microphone. Human entered data is a common source of error, and often leads to a poor experience on the side of the end user.

5 Approach

By using an RFID reader near the presenter location, the speaker can effectively be identified. This eliminates the need for human interaction to send out the name of the current speaker. Identification information can then be distributed through established communication channels, such as HTTP or Text Message. By providing an extensible platform for distribution, all participants of a large conference can receive relevant alerts regarding the talks in which they are interested.

The RFIDentify platform is initially provided as a set of *client/server* programs. The *client* refers to a device handling RFID hardware. We envision it to be a small device running an embedded version of Linux, perhaps attached to the microphone stand on the stage. This client will communicate with the *server*, which is tasked with performing some action in response to each new identification. For example, the server may display a new set of slides upon the recognition of a new speaker, or dispatch text messages to all conference attendees.

5.1 Concerns

The use of a RFID tag can raise security concerns from the users. RFID readers have been shown to be be able to discover tags from few meters away. This can be alleviated with use of a shielding device that blocks RFID frequencies from being intercepted by unwanted readers. Another strategy is to have low frequency card that only have an unique ID associated with the card. A database would then store the information of the card holder. If anyone reads the card, they will not be able to gather any useful information about the person. We take the latter approach when considering the design of the RFIDentify system.

5.2 Server

The server Stores a database of RFID tag ID numbers associated with each presenter. The database allows quick access to the speaker's contact information, such as their email address or the location of their presentation. It would be possible (and encouraged!) to further extend this database to track other personal information. As an example, the ID cards might also serve as meal tickets, which could be swiped at the checkout counter of participating conference restaurants. An *available-meals* attribute could be appended to the database, and linked by primary key.

5.2.1 Dependencies

1. **Network** - The Server requires a network connection to receive incoming presence responses from RFID readers.

2. **Java** - The Server is implemented and packaged as a Java Archive, targetting all major platforms.

5.3 Client

5.3.1 Dependencies

1. **Linux 2.6** - The client software targets the Linux operating system. In particular, the RFIDentify client targets embedded Linux systems with limited resources.

2. **RFID reader** - The client must be connected to a RFID device capable of collecting the unique ID of each card. Supporting libraries are also required to drive this specific hardware.

3. **WiFi connection** - It is envisioned that the client runs on a small and mobile device, connected to its server via a wireless connection. Service discovery is included, so further network configuration is not necessary.

4. **Shared Libraries** - The client executable relies upon a few common shared libraries: PThreads, cURL, and Avahi.

5.3.2 Service Discovery

The RFIDentify system employs mDNS service discovery to coordinate client/server communication. RFIDentify servers emit particular service notifications, which are recorded by clients. When an RFID tag is identified, a client will automatically notify each server via an HTTP GET request. In other words, the system "just works" out of the box.

5.3.3 Configuration

Management of settings specific to unique clients can be accomplished through a configuration file. By default, this file is to be found in the current executing directory, with the filename ".rfid_client.conf". Additional hardcoded servers can be specified, for example. mDNS settings are also available for modification.

5.3.4 Implementation

The client is implemented in the C programming language, targetting the Linux platform. It is compiled in to a single binary file, to be executed suitable permissions (particularly accessing USB or similar devices). After setup, the becomes multi-threaded with two main loops. One loop interfaces with a standard Avahi mDNS stack, and updates available server information. As RFIDentify servers are added or removed from the network, this loop ensures that only reachable servers receive identification events. The second loop handles RFID hardware interfaces, and periodically polls for new identifications. When an event occurs, a callback handler is invoked. The current implementation employs the cURL HTTP library to pass the newly encountered ID to all available servers.

6 Technologies

The client and server are implementing using a number of open source toolkits and software. By reusing public code and APIs, the implementation limits possible bugs, and is forced to comply with standard design patterns.

6.1 Server

1. **Apache** - Server core implementation for handling client requests. The httpcore.jar used in the implementation handles low level functionality of client handling from opening ports to passing the requests onto higher level HTTP request handlers.

2. **SIMPLE** - SIP message passing implementation. Used to alert a presence server that a current speaker is presenting.

6.2 Client

1. **Avahi** - Avahi is a free Zeroconf implementation, including a system for multicast DNS/DNS-SD service discovery. It allows programs to publish and discover services and hosts running on a local network with no specific configuration.

2. **PThreads** - Pthreads is a POSIX standard for threads defined in *POSIX.1c*. Implementations of the API are available on many Unix-like POSIX systems such as FreeBSD, NetBSD, GNU/Linux, Mac OS X and Solaris.

3. **cURL** - cURL is an easy-to-use client-side URL transfer library supporting FTP, FTPS, HTTP, HTTPS, SCP, and other formats.

6.3 Hardware

1. **DLP RFID1** - The DLP-RFID1 RFID Reader/Writer is an inexpensive RFID device targetting the 13.56 MHz frequency. It communicates via a USB connection and FTDI protocol. However, the device specific commands have not been released to the public. For the purposes of the RFIDentify project, the RFID poll command was reverse engineered through a binary analysis of the Microsoft Windows dynamic link library. The DLP RFID1 is fully supported for RFIDentify.

2. **FTDI/Big Reader** - The FTDI/Big Reader (term coined for this project) was developed by Marc Petit-Huguenin to address inefficiencies at IETF conferences. The RFID reader communicates via a serial protocol. The current implementation does not yet fully support this RFID reader.

7 Acknowledgements

We would like to specially acknowledge Professor Henning Schulzrinne at Columbia University for guiding us through the project. His inspiration and ideas were critical in the final design and implementation.

We would also like to thank Omer Boyaci for providing insight into the SECE project currently underway at Columbia University.