Smartphone Packet Relay Networks

David Flanagan
Rochester Institute of Technology
August 2012
Outline

• Project Overview
• Introduction/Motivations
• Related Work
• Implementation Hardware
• Project Architecture
• Results
• Future Work
• Conclusion
• Questions
SPRN Overview

• SPRN: Smartphone Packet Relay Network
• Ad-hoc message relaying, phone to phone
• Self forming, Multi-hop
• Bluetooth, initially
• On-demand routing
• Backup for primary cellular network
Motivation

- Backup communication for normal cell network
  - SPRN activates when there is a cell outage
  - Many scenarios for cell outages:
    - Earthquakes, fires, overloaded bandwidth, terrorist attacks, etc.
  - Backup communication eases relief efforts
    - Evacuation instructions
    - Medical response
    - Contacting family/friends
Motivation

In 2005 Hurricane Katrina caused widespread death and destruction.

Over 1,400 cellphone towers were made inoperable.

3 Million customers were without phone lines.

Existing infrastructure and communications strategies were rendered useless\(^8\).

Having a system like SPRN would have helped alleviate communication issues.
Motivation

• Additional Motivations
  ▫ Expanded cell radius
  ▫ Bypass use of cell towers altogether
Background and Related Work
Related Work

- **PRNET (DARPA, 1973)**\(^3\)
  - Ad hoc radio network with special hardware
  - Self forming
  - Addressed many issues: node topology, routing, packet loss, signal interference, etc.

- **Multihop Cellular Network (Ying-Dar Lin 1999)**\(^5,6\)
  - Multi-hops across phones to get to cellular backbone
  - Increase of range and throughput
Related Work

  - Finds most efficient path to cellular backbone
  - Uses one or more ad hoc node hops
  - Goal: Best service with highest efficiency for a node

- **UCAN: Unified Cellular and Ad hoc Network (2003)**[2]
  - Provides 3G speeds to older devices (without 3G)
  - 3G phones are used as proxies
  - Older phones connect to proxy via wi-fi

- **Serval Project (2011)**[1]
  - Ad hoc mesh networking, where no infrastructure exists
  - 'Permanent' ad hoc network
Related Work

• Our Contributions
  ▫ Cellphone communication bypassing infrastructure completely
  ▫ Utilization of Android smartphones
    • 'Cheap' unspecialized hardware
Architecture

Hardware | Operation | Architecture Detail
Hardware

- Android smartphone running v2.2 (Froyo)
  - Bluetooth
- Root access to operating system
  - Less portable, but more application control
- Android OS very similar to Linux
- Dalvik VM, runs Java code
SPRN Operation

Ideal operation of SPRN is as follows:

- SPRN inactive, runs in background, low detection frequency
- Detects low signal strength, SPRN activates, high detection frequency
- Devices locate all neighbors in range (discovery)
- A packet needs to be transferred
  - Route requested and formed on demand (if possible)
  - Message passed along route (one or more hops)
Architecture: Overview

- 3-layer model: Datalink, Routing, Application
  - More flexibility, especially for future enhancements
Datalink Layer

Overview | Data Structures | Operation | Communication Procedures
**Architecture: Datalink**

- Written in C
- Runs as Root, background daemon
- Handles:
  - Physical communication (Bluetooth v2.1 + EDR)
  - Node Association/Disassociation (Discovery)
- Communicates with Routing layer via socket
- Datalink specification
  - Packet formats, communication procedures
  - Abstraction of intra-node communication and inter-node communication
  - Nodes are not required to be Android phones!
Datalink Data Structures

- RemoteDeviceInfo
  - One per device ("node")
  - Information necessary to communicate
  - Cached for faster lookups after initial contact

<table>
<thead>
<tr>
<th>RemoteDeviceInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID: unsigned 64 bit</td>
</tr>
<tr>
<td>phoneID: unsigned 64 bit</td>
</tr>
<tr>
<td>versionNum: unsigned 32 bit</td>
</tr>
<tr>
<td>isValid: 8 bit</td>
</tr>
<tr>
<td>RSSI: unsigned 8 bit</td>
</tr>
<tr>
<td>BD ADDR: 6 bytes</td>
</tr>
</tbody>
</table>
Datalink Data Structures

- **ActiveConnection**
  - One thread per device link
    - Node address is primary key
  - Eliminates contention and multiple device links
  - SynchronizedThreadQueue
    - Communication between threads and/or layers

```c
typedef struct {
    RemoteDeviceInfo remoteDevInfo;
    SynchronizedThreadMessageQueue outgoingMsgs;
    unsigned channel;
    bool isActive;
    bool isReserved;
    pthread_mutex_t mutex;
} ActiveConnection;
```
Datalink Operation

- Commands via L2 socket
- Incoming connections from remote devices
**Datalink: Discovery**

- Most aspects handled by Bluetooth hardware
- Inquiry interval and duration are configurable
- Event Sequence:
  - BT HCI reports devices in range (and signal strength)
  - Devices are either known already or new
  - New devices have preliminary communication: “version request”
  - L2 is updated with active node list
**Datalink: Discovery**

- **Version Request**
  - Determines if devices are compatible
  - Exchanges software version and phone ID

- **Packet format**
  - Request and Reply
  - Same format for each (different header type though)

![Packet format diagram](image-url)
**Datalink: Packets**

- **Two Varieties**
  - Inter-node and intra-node

- **Inter-node**
  - “Send Message”
    - Upper layer data sent to remote device

**Header:**

- Addresses are BT hardware addresses
Routing Layer

Overview | Routing Protocol | Operation | Data Structures
Architecture: Routing

• Written in C

• Runs in a Native Thread
  - Interacts with L3 via JNI calls
  - Interacts with L1 via socket

• The Brain of SPRN
  - Keeps knowledge of network routes and states
  - Passes data between layers
Routing: AODV Protocol

- Ad hoc On-demand Distance Vector (AODV) routing
  - Specified by RFC 3561\textsuperscript{[7]}
- On Demand
  - Routes only formed when needed
  - Low Maintenance
  - Less routing information passed remotely
  - Quick adaptation to new topology
  - No routing loops
- SPRN uses AODV, with minor modifications
  - Simplified, adapted for our network protocol
Routing: AODV Protocol

- Two main control packets: Route Request (RREQ) and Route Reply (RREP)

- RREQ sent out (broadcast) on demand
  - Search for a destination
  - May be received by end node or intermediate node with path to destination

- RREP sent out (unicast) to RREQ sender
  - Destination has been found
  - Takes reverse path of RREQ
    - Route is 'established' by each node along this path
Routing: AODV Protocol

RREQ packet format:

RREP packet format:
Routing: Operation

- Main Thread loops over all function calls; sleeps
- Data is buffered until it can be processed
Routing: Data Structures

• Route
  - Holds information needed to get one hop closer to destination
  - One route per destination; storage in routing table

<table>
<thead>
<tr>
<th>Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID : unsigned 64 bit</td>
</tr>
<tr>
<td>destID : unsigned 64 bit</td>
</tr>
<tr>
<td>nextHop : unsigned 64 bit</td>
</tr>
<tr>
<td>dstSeqNum : unsigned 32 bit</td>
</tr>
<tr>
<td>hopCount : unsigned 32 bit</td>
</tr>
<tr>
<td>lifetime : unsigned 32 bit</td>
</tr>
<tr>
<td>dstValid : unsigned 8 bit</td>
</tr>
<tr>
<td>flags : unsigned 8 bit</td>
</tr>
</tbody>
</table>

• Higher sequence numbers indicate newer routes
Routing: Data Structures

• Node
  - Similar to RemoteDeviceInfo in L1; one per node
  - Holds data needed to convert between address formats
    - L2 deals with phone IDs (phone numbers), L1 deals with Bluetooth addresses

<table>
<thead>
<tr>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID : unsigned 64 bit</td>
</tr>
<tr>
<td>errCnt : unsigned 16 bit</td>
</tr>
<tr>
<td>hwAddr : 6 bytes</td>
</tr>
<tr>
<td>RSSI : 8 bits</td>
</tr>
<tr>
<td>flags : unsigned 8 bits</td>
</tr>
</tbody>
</table>
Application Layer

Overview | User Interface | Operation | Actual vs. Ideal Design
Architecture: Application

- Written in Java (uses Android SDK)
- Interacts with L2 through JNI calls
- Handles
  - User input
  - Displays nearby nodes
  - Activates SPRN (not implemented)
Application Interface

SPRN
My phone number is 5855066000
Destination Phone Number

<table>
<thead>
<tr>
<th>Send</th>
<th>Clear Output</th>
</tr>
</thead>
</table>

Result
Received: This is a test message (from 5855066003)

<table>
<thead>
<tr>
<th>5855066001</th>
<th>RSSI: -20</th>
</tr>
</thead>
<tbody>
<tr>
<td>5855066003</td>
<td>RSSI: -33</td>
</tr>
</tbody>
</table>
Application: Operation

- Initialization
  - Read in properties (external file)
  - Start routing layer
  - Initialize GUI components

- Main execution thread
  - Regularly polls L2 via JNI for incoming data
  - Processing incoming data
  - Processes user requests (send data)
Application: Operation

• Device ID
  • Device's phone number (e.g. 5855066000)
  • Used to direct messages to an endpoint
  • Set automatically from phone or by user input
    • Implemented as a property
  • Removes need for central authority
Application: Actual Vs. Ideal

- Actual implementation does not fit ideal scenario
- Ideally, user interface is transparent
  - Automatic device pairing
  - Automatic SPRN activation
  - Automatic intercept and reroute of data
- Current Android platform limits options
  - Automatic SPRN activation is possible
  - Device pairing is semi-transparent
  - Intercepting (app) data is not possible
Results

Range | Interference | Latency | Testing Limitations
Results: Distance Test

- Bluetooth range is short!
- Signal strength falls off logarithmically with distance
- 30 feet between devices; approximate limit
  - Can go further, more transmission errors though
Results: Interference Test

- Devices placed in a circle of variable, expanding radius
- Further distances created more dropped connections
  - 4 devices within an 8 foot radius were unable to maintain all connections
- Interference mostly caused by BT discovery?
  - Problem may be avoided by using different technology
- Node density per area must be limited to avoid interference
Results: Latency Test

• Main metric, defines network scalability
• Message sending time recorded vs. number of hops
  □ Initial route creation and traversal time
  □ Additional message traversal time

<table>
<thead>
<tr>
<th>#Hops</th>
<th>Initial (sec)</th>
<th>Additional (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21.6</td>
<td>4.1</td>
</tr>
<tr>
<td>2</td>
<td>38.3</td>
<td>8.3</td>
</tr>
<tr>
<td>3</td>
<td>46.0</td>
<td>11.0</td>
</tr>
</tbody>
</table>

• Limited data, but trend shows poor scalability
  □ Would take about 25 minutes to traverse 100 nodes
Results: Latency Test

• High latency due mainly to slow discovery

• BT Inquiry takes about 10 seconds at each node, not including processing overhead

• Additional messages are faster, but still slow

• Different wireless technology may change scalability
Results: Testing Limitations

• Need to test: power consumption
  □ Obvious concern, but difficult to gauge
  □ Condition of battery?
  □ Running background programs?

• Limited testing hardware
  □ Tested with 4 devices; more data needed for better conclusions
Future Work

Security | Software Optimizations | Hardware Optimizations
Future Work: Security

• Major concern!

• Root privileges
  - Presents more security risk than necessary(?)

• AODV, no implicit security
  - Would need to add authentication, maybe encryption
  - Protect from fake RREP or RERR messages

• ID generation
  - User controlled, easy to forge/impersonate

• Unfair use of network
  - Crediting for usage?
Future Work: Software Optimization

• No optimizations yet
  □ Need to establish baseline performance

• Finding the most efficient route
  □ Factors (per hop): signal strength, bandwidth, velocity, transmission power
  □ More information = better routes, but greater overhead

• Packet optimization
  □ Alignment, field efficiency, etc

• General Code optimization
Future Work: Hardware Optimization

- Mainly focused on not using Bluetooth
  - Could decrease initial latency with faster discovery
  - Increase range with higher power transmitter
  - Decrease interference with better multiplexing technology
Conclusion
**Conclusion**

- SPRN is a prototype system
- Foundation provided for future work
- Android is *almost* ideal for SPRN
  - Needs some modifications to allow more 'risky' behavior
- Scalability is an issue currently
- Bluetooth is not ideal for this application
  - High latency, short range
  - Good for prototype development though
References


