Sony Ericsson Z525a Bluetooth enabled mobile phone.

Bluetooth Triangulator

CSE 237A Final Project – Fall 2006

- Varun Almaula and David Cheng
Outline

- Project Overview and Motivation
- Related Work – Previous Projects
- Known Issues
- Project Description
- Software Installation and Hardware Logistics
- Physical Layout
- Project Challenges and Issues
- Results
- Project Conclusions
  - Methods for Improvement
  - Future applications and outstanding issues
- Summary
- References
- Q & A
Project Overview and Motivation

- Location schemes for wireless protocols allow for an entirely new class of high level software applications
  - Location Based Services
    - Real time mobile information
    - Mobile billing services
    - Emergency services
    - Tracking
  - Ubiquitous Computing
    - Social computing

- Use already widespread wireless infrastructures to provide all necessary hardware resources
  - GPS/satellite
  - Cellular network
  - 802.11 WiFi
  - Bluetooth
Project Overview and Motivation – LBS
Project Overview and Motivation – Ubiquitous Computing

- UCSD ActiveCampus project lead by professor Griswold
• Can we extend this type of location service to Bluetooth?

• Why would we want to locate with Bluetooth?
  
  – Very interesting high level software applications can be built on top of a Bluetooth locator utility

  • Social networking applications for mobile phone users

  • Security: locating unknown/unauthorized Bluetooth devices in a highly data sensitive government network
Previous Projects – Known Issues

• What can we learn from the previous projects?
  – Bluetooth location is not that precise
    • RSSI readings vary greatly for many reasons
      – Environmental conditions and physical objects create RF interference
      – HW manufacturers of Bluetooth devices may not produce accurate RSSI measurements
    – For best results, the system should be contoured for the room/environment
      • Hard code the system HW & SW for the room
  – Advanced AP hardware can improve the accuracy of RSSI readings
  – The Bluetooth discovery process may not always find all Bluetooth devices in the area
    • Bluetooth uses frequency hopping and changes channels when sending outgoing discover messages
Software Installation

- On an x86 Red Hat machine, recompiled the Linux kernel with Bluetooth stack for our target ARM architecture
  - Used ‘menu-config’ and selected appropriate Bluetooth options
- Re-flashed the new cross-compiled Linux OS to the target
- The x86 machine already had Bluetooth networking support as part of the Red Hat Enterprise release
- Download the BlueZ package source
  - Compiled and installed BlueZ libraries on x86 machine
  - Compiled and installed BlueZ utilities on x86 machine
  - Cross-compiled for ARM architecture and installed BlueZ libraries to mounted directory
  - Cross-compiled for ARM architecture and installed BlueZ utilities to mounted directory
  - Used BlueZ version 2.25 for reasons described later
Hardware Installation

- Master server node and all processing done on Intel XScale PXA27x Developer Board
- Use 3 USB Bluetooth dongles as the master nodes
  - 1 D–Link dongle used for XScale platform
  - 1 D–Link AND 1 Zonet dongle used on a single x86 machine simulating two different clients
- Use many USB extension cables to place the two client master nodes in distinct locations
Hardware Installation

- Used mobile phone and laptop as test Bluetooth devices
Project Challenges and Issues

- Cross compiling latest BlueZ (3.3.7) was unsuccessful
  - Had a dependency on ‘dbus’ which did not seem to be cross-compile friendly
  - Learned that BlueZ package created a dependency on dbus on version 3.x
  - Simply ‘downgraded’ to version 2.25 and install was a breeze

- Discovery issues
  - Devices within range may not be discovered
    • Why? Frequency hopping
    • Why? Devices may not be listening
  - Can take a while
    • Why? 10s by default
    • Shorten discovery period → More devices may be left out
Project Challenges and Issues

- Inquiry mode issues
  - Two methods of obtaining RSSI information
    - Create a connection to the Bluetooth device and then read RSSI
    - Obtain RSSI information during an inquiry without connection overhead
      - Change inquiry mode to ‘inquiry with RSSI’
      - This needs to be supported by the HW on both the sender and receiver
      - Was difficult to obtain correct results
      - Decided to go with first method for ease of implementation at this point
Project Challenges and Issues

- Creating connection issues
  - Failed to create connection
    - Why? Connection request may not be granted
    - Why? Resources are limited
    - Why? Page timeout
      - Huh?
      - Too many people paging me!
  - Cannot have a cake and eat it too
    - Cannot do discovery while having a connection
      - Why? We do not know. This should not have happened
      - Workaround – Tear down connection when start discovering
Project Challenges and Issues

- Link maintenance and RSSI collection issues
  - Tear down connections which are no longer needed
    - Resources conservation
      - Why? Resources are very limited for embedded systems
    - Reduce the load on the target device (slave)
      - Why? Easier for creating another connection to it
  - RSSI readings fluctuate frequently and easily
    - Need to collect the readings over a period of time
    - Any obstacle in the way can alter the RSSI reading greatly
      - Nothing we can do about this

- Tearing down connection issues
  - Do not create another connection while one is being torn down
    - Why? Bluetooth, by nature, is NOT high speed. Layers. Wireless network
  - Do not start another task while a connection is being torn down
    - Why? See above
Results

• Re-wrote the hcitool utility for our application: one version for the client nodes and one version for the server node

• Each Client:
  – Setup sockets for data communication to server → Scan for devices → Obtain RSSI values for each device → Pack information into message → Write to socket

<table>
<thead>
<tr>
<th>Client Message Packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

• Server
  – Setup sockets for data communication to clients → Scan for devices → Obtain RSSI values for each device → Store information locally → Listen on socket for client messages → Process each client’s data → Calculate and print distances
Results

• RSSI→distance mapping done empirically (it is not linear)

<table>
<thead>
<tr>
<th>RSSI (dBm)</th>
<th>Distance (feet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>8</td>
</tr>
<tr>
<td>-2</td>
<td>12</td>
</tr>
<tr>
<td>-3</td>
<td>16</td>
</tr>
<tr>
<td>-4</td>
<td>20</td>
</tr>
<tr>
<td>-5</td>
<td>24</td>
</tr>
<tr>
<td>-6 to -9</td>
<td>28</td>
</tr>
<tr>
<td>-10</td>
<td>Not connected/outside room</td>
</tr>
</tbody>
</table>

• Use simple trilateration scheme to calculate coordinates and distance

• Location is done in ‘zones’
  - Within 8 – 10 feet – strong signal strength
  - +/- 5 feet accuracy 90% of the time at 17 feet
  - +/- 5 feet accuracy 80% of the time at 30 feet
Project Conclusions – Methods for Improvement

- Change inquiry method to remove connection overhead
  - Do discovery less frequently
  - Make the inquiry with RSSI mode work

- Use USB Bluetooth dongles with accurate RSSI readings

- Obtain more hardware to place USB dongles in any location in the room (perhaps on ceiling corners)

- Use advanced Bluetooth AP’s with multiple high gain antennas with variable attenuators
Project Conclusions – Future Outlook

• Outstanding issues

  - Update frequency
    • Tradeoff between processing overhead and ‘real-time’ accuracy
    • Handling moving Bluetooth objects
    • Scalability for large numbers of Bluetooth devices

  - Distance accuracy for different classes of Bluetooth devices (1–100m ranges)

  - Create more wireless friendly ‘smart spaces’

  - Standardize HW RSSI accuracy requirements

  - Perhaps the next version of Bluetooth could include a basic interface for obtaining positioning information
Summary

• Project Overview and Motivation
• Related Work – Previous Projects
• Known Issues

• Project Description
• Software Installation and Hardware Logistics
• Physical Layout
• Project Challenges and Issues
• Results

• Project Conclusions
  – Methods for Improvement
  – Future applications and outstanding issues

• Summary
• References
• Q & A
References

- Bluetooth SIG. “Specification of the Bluetooth System v2.0 Volume 0, Master Table of Contents and Compliance Requirements” 2004.
• Thank you

• Questions?