Beetle:

Operating System Support for the Internet of Things

Amit Levy, James Hong, Laurynas Riliskis, Philip Levis, David Mazières, and Keith Winstein
The Internet of Things *Ideal Future*
The Internet of Things Today
It's Not An Internet

- Connectivity is poor and constrained
  - Edge devices cannot communicate with each other
  - Edge devices can only communicate with one application
- Vertical integration of peripherals, gateways, and cloud
- Simple, desirable use cases are impossible
  - Monitor battery life of all my devices
Why? Some non-Fundamental Reasons

• Vendors want to “own the user experience”
• Standardization takes time
  – WWW, Z-Wave positive historical examples
  – Messaging probably a negative example
What are the *Fundamental* Reasons?
It's the gateway stupid!
“It's the [Gateway] Stupid!”

- IoT devices are feable and weak
  - Low power: achieved by being mostly off
  - Simple network protocols: Bluetooth Low Energy et al
- Naming is harder
  - “Honey, what's the toaster's IP address, again?”

It's up to gateways to provide: connectivity, naming, security...
Operating Systems Today
Operating Systems Today
Operating Systems Today

- Small number of tailored device interfaces
  - Printer
  - File system
  - X Windows
  - HID
  - TCP/IP stack
Linux is not IoT Scale!
Operating Systems Today

- The IoT will have 1,000,000...,000 device types!
- We can't hope to support each type in every OS separately
- For now, instead of safe, shared interfaces we get exclusive streams
  - L2CAP sockets for BLE, serial devices for ZigBee, etc
  - 6lowpan is an exception (sort of)
We need a single OS interface for all devices!
Outline

- Looooooong Intro
- Bluetooth LE architecture
- Beetle
  - Network architecture
  - Mechanisms:
    - HAT
    - Virtual devices
    - Service export control
Outline

• Looooooong Intro
• Bluetooth LE architecture
• Beetle
  - Network architecture
  - Mechanisms:
    • HAT
    • Virtual devices
    • Service export control
Bluetooth Low Energy

• Single-hop protocol
• Physical, Link and Application layers
• Optimized for small exchanges and low energy:
  – ~24 byte exchanges; infrequently
  – μA power consumption
  – Can run for years on coin battery
Bluetooth Low Energy

- Advertising Packets
- Generic Attribute Protocol (GATT)
  - L2CAP
- Link Layers
- Connections
Link Layer

• “Piconet” topology

• Two roles:
  – Peripheral (fitness band, watch, dead-bolt, etc)
  – Central (smart phone, laptop, gateway, etc)

• Centrals manage connections with multiple peripherals

• Peripherals can connect to a single central only
L2CAP Channels

- Logical channels over single link
- Reliable
- Some channels reserved (e.g. GATT, signaling)
**Generic Attribute Protocol (GATT)**

<table>
<thead>
<tr>
<th>Handle</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0200</td>
<td>service</td>
<td>glucose</td>
</tr>
<tr>
<td>0x0201</td>
<td>measurement</td>
<td>data</td>
</tr>
<tr>
<td>0x0202</td>
<td>context</td>
<td>data</td>
</tr>
<tr>
<td>0x0203</td>
<td>feature</td>
<td>data</td>
</tr>
<tr>
<td>0x0204</td>
<td>record control</td>
<td>data</td>
</tr>
<tr>
<td>0x0205</td>
<td>service</td>
<td>time</td>
</tr>
</tbody>
</table>

**Characteristics**

<table>
<thead>
<tr>
<th>Handle</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0533</td>
<td>service</td>
<td>heart rate</td>
</tr>
<tr>
<td>0x0535</td>
<td>measurement</td>
<td>data</td>
</tr>
<tr>
<td>0x0540</td>
<td>location</td>
<td>data</td>
</tr>
</tbody>
</table>
GATT

- Two roles:
  - Server has the attributes
  - Peripherals and Centrals can be both clients and servers simultaneously

- Key/Type/Value store:
  - Read/Write
  - Notify/Indicate
  - Find by type

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Handle</th>
<th>Opcode parameters (type, value ...)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GATT: Simple Example

Notify 0x7: 152bpm
Notify every 1 second
Notify 0x7: 154bpm
Notify 0x7: 152bpm
Notify 0x7: 157bpm

Server

Client
A peripheral can only maintain one open connection!
One-to-One Communication

- App
- App
- App
- Gateway
- OS
- BLE
The Internet of Things *Today*
Today: Gateway Interposes on Data

- Each peripheral connects to a single app on the gateway
  - Can only communicate directly with that app
- App consumes GATT data. Mediates only supported interactions:
  - Issue GATT commands to other connected peripherals
  - Proprietary protocol to servers (e.g. over app-specific HTTP)
  - (Limited) Intent-based interface to other apps
- The app doesn't support an interaction you want?
  - Tough luck...
GATT: Three Important Properties

• Self-Describing:
  – Standardized service/characteristic types
  – Incorporates service discovery

• Transactional
  – Only one outstanding command per connections in each direction

• High level
  – Application-level transactions == protocol transactions
Outline

• Loooooooong Intro
• Bluetooth LE architecture
• Beetle
  – Network architecture
  – Mechanisms:
    • HAT
    • Virtual devices
    • Service export control
Beetle

- Builds a *network* out of BLE
- A software layer that runs on your gateway
  - Device-to-device communication
  - *Safe* multi-app communication, locally or over Internet
  - Fine grained access control
- Completely backwards compatible with existing BLE devices
Beetle: Design Overview

- Privileged user-space process on Linux and Android
- Provides networking to BLE devices
- Gateway routes between peripherals, apps and cloud
- Leverage richer user-interface on gateway
Beetle: Gateway Mechanisms

• Handle address translation (HAT)
  – Multi-link networking

• Virtual devices
  – Software connectivity
  – Interface with other protocols (e.g. HTTP, Intents)

• Service export control
  – Fined-grained security policies
  – Naming
Handle Address Translation (HAT)

- Re-export peripheral services as gateway services
- Proxied attributes on the gateway
  - Associated with a remote attribute on a peripheral
  - Beetle routes messages to proxied attributes to the appropriate peripherals
- Translate peripherals handles into gateway address space
  - Similar role to NAT in TCP/IP world
Handle Address Translation (HAT)
HAT: Discovery

• Typical BLE connection has fixed set of services
• In Beetle, new services appear as more peripherals connect or policy is changed
• Take advantage of “Service Changed” characteristic
  – Notifies client when new set of services changes
  – Provides a range of affected handles
• Keep track of which peripherals might notice the service has changed to minimize noise
  – If a peripheral never asks for a service, it shouldn't matter
HAT: Notifications

- GATT notifications are a two-step process:
  - *Subscribe/unsubscribe* to notification by writing 1 or 0 to an attribute
  - Server begins notifying when value changes

- Cannot re-expose subscription attribute directly

- Instead:
  - Maintain a subscription set for every server notification source
  - Intercept *subscribe* and *unsubscribe* messages
  - Only forward first *subscribe* or last *unsubscribe* to server
HAT Creates a *Network*

- Re-exporting attributes on gateway enables peripheral-to-peripheral communication
- Aggregating attributes from multiple servers allows many-to-many peripheral communication
- HAT must maintain app-level protocol semantic
- Leverage knowledge of app-level protocol semantics to retain reasonable performance
Demo: Lights and Switches
Virtual Devices

• Virtual devices speak GATT for non-BLE links:
  – IPC, TCP/IP, USB, etc

• Provide access to non BLE services
  – GPS
  – Emulated device with test data
  – Legacy Internet services (e.g. HTTP)

• Complexity handled by HAT
Virtual Devices: Local

- A user-level process that speaks GATT
- Access to Beetle over IPC (e.g. UNIX domain sockets)
- Similar to programming an app now (identical on Android)
- Very useful:
  - Multiple user apps
  - Expose local, non-BLE, sensors
  - Prototyping hardware
  - Custom multiplexing
Virtual Devices: Network Services

• Virtual devices can exist on the Internet
  – In the cloud, local area network
• Scenario 1: Internet service supports Beetle
  – Beetle OS service connects directly over TCP
  – Don't need to write a tailored app
• Scenario 2: Legacy Internet service (e.g. HTTP/REST)
  – A local virtual device exports data over the legacy protocol
Service Export Control

• So much connectivity!!
• Need a way to control who sees what
  − Strava shouldn't only see my current heart rate when I allow it
• Routing at app-level protocol gives us more flexibility
• Many possible criteria for access control
  − Physical location
  − Identity
  − Pre-established trust
  − Out-of-band authentication (e.g. user login)
Beetle

- Gateway should route communication but not mediate application data
- Beetle is an OS service on the gateway that creates a network from BLE
- Three key mechanisms:
  - HAT for peripheral communication
  - Virtual devices for multiple-apps, device emulation and connecting other networks
  - Service export control pushes policies to more featureful gateway devices
- Completely backwards compatible with existing BLE devices
Questions?
Beetle Linux Implementation

- Linux user-level process
- ~1300 lines of code (in Go)
- No changes to Kernel
  - Although could be useful
- Global handle address space
- Virtual devices over UNIX domain sockets
Peripheral-to-Peripheral RTT

![Graph showing the relationship between connection interval and round-trip latency forthrough proxy and direct connection.](image-url)