QtBluetooth on Mobile Devices
A Dragon Guide

Mathias Hasselmann, Senior Software Engineer at KDAB
About me

• developing Linux software for almost 20 years now
• implemented central components for Nokia’s Mæmo and Meego phones
• various customer projects in mobile and embedded with KDAB since 2013
Topics

• Mobile Platforms
• Short overview on Bluetooth
• Device and service discovery
• Transport protocol
Mobile Platforms

• Gartner reports for Q4 of 2016:
  - Android: 81.7%
  - iOS: 17.9%
  - Others: 0.4%

→ just buy your customer an iPhone X – more profitable (than to support their other platform)

• huge variety of devices
• no control over specifications
Bluetooth

• actively developed since 1999
• shares 2.4 GHz with WiFi, ovens, and fridges
• huge specification
• countless profiles
• many implementations, more or less interoperable
How about WiFi instead?

• very reliable, efficient, low latency
• major issues:
  – restricted APIs for network discovery
  – missing APIs for automatic network selection
  – most Importantly: What about Internet?
Bluetooth Classic vs. Low Energy

• Classic Bluetooth
  - successful for headphones, in-car entertainment, hands-free system
  - way too inefficient for wearable gadgets

• BTLE allows days instead of hours
  - more reasonable timings
  - much simpler protocols

• no backwards compatibility
• much slower
Device Discovery
BluetoothDeviceDiscoveryAgent

• first results are cached, usually can be told from RSSI
• RSSI highly hardware specific – useless for proximity estimation
• stacks often report classic and BTLE devices – independent of selected discovery mechanism
• reported core configuration in QBluetoothDeviceInfo is unreliable
• spurious results from incomplete BTLE beacons: “Mathias’ awesome mobile gadget”
Service Discovery
QBluetoothServiceDiscoveryAgent

• traditionally via UUID in SDP record
• “everything is a serial port”
  - generic SDP record with SPP UUID
  - custom record with product specific UUID
• Android phones report:
  - all SDP records
  - only the first record they see
  - only the last record they see
  - only the standard records they see
Service Discovery
QBluetoothServiceDiscoveryAgent

• SDP just doesn’t work well enough on Android
• Hardware address
  - controlled by Bluetooth chip vendor
  - not accessible on iOS
• Bluetooth device name
  - up to 255 characters in UTF-8
  - cache and protocol issues
• Generic Attributes (GATT)
Transport Protocol
QBluetoothSocket, RFCOMM

• API level zoo for Android version of QtBluetooth
• some Android versions required SDP to create socket
  - which just is highly unreliable (on Android) as we learned
  - no public API to selected fixed channel
  - had to patch QtBluetooth to use fixed channel (Qt Commercial)
• iOS:
  - requires special crypto chip and MFi license from Apple
  - underlying iAP2 protocol not supported by QtBluetooth
Transport Protocol
Bluetooth Low Energy

• luckily BTLE is well supported both by Android and iOS
• serial port emulation via GATT
• almost transparent for µ-controllers
• very cheap controllers from China (“HM-10”)
• much slower than real SPP via RFCOMM:
  - GATT attribute abused as USART buffer (MTU 20)
  - confirmation packets after every 20 bytes, or strict timing and custom transport security layer
• sometimes flow control via separate GATT attribute
Transport Protocol
Generic Attributes (GATT)

- generic attribute protocol
- triple based: service UUID, attribute UUID, value
- very similar to RDF ontologies*)
- (usually) trivial to map to hardware state
- avoids overhead of custom protocols (transport safety, multiplexer, control)

*) “They call us crazy, but we store Contacts in Tracker” – Desktop Summit 2011
Thank you!*)

www.kdab.com

mathias.hasselmann@kdab.com

*) ...and to the fine people sharing their pretty dragon pictures on pixabay.com