Peer-to-Peer Technology: Driving Innovative User Experiences in Mobile

Marcello Lioy
Director, Engineering, Qualcomm Innovation Center, Inc.
Agenda

- AllJoyn Overview
- Architecture and Security Concepts
- Performance Considerations
- Open Source
- Q&A
AllJoyn Overview
What is AllJoyn?

AllJoyn enables ad hoc, proximity based, peer-to-peer, bearer agnostic networking between devices and applications.
What is peer-to-peer and why is it important?

- Enables devices and applications to communicate with each other without the need of a server
- Mobile peer-to-peer will enable a new generation of proximal experiences
Why Peer-to-Peer (P2P) Is Hard

Radio technologies alone have not made P2P easy. (ex: Bluetooth® pairing)
P2P enablers have not become ubiquitous or open
P2P via the cloud is not proximal, adds latency, requires a WWAN, and subjects the user to data charges
AllJoyn Makes Peer-to-Peer Simple

**Discover**
devices and applications around you

**Adapt**
to devices coming and going

**Manage**
transports like Bluetooth® and Wi-Fi and message routing across them

**Interoperate**
across disparate Operating Systems and bearers

**Exchange**
Information in a secure manner
What new experiences can AllJoyn enable?
Architecture and Security Concepts
Overview

- AllJoyn is a distributed software bus
  - Each device runs a bus daemon
  - Applications communicate directly with daemon
  - Daemons on each device communicate with daemons on other devices
  - Daemons do message routing and namespace management

- Bus formation is ad hoc
  - Based on proximal discovery
  - Abstracts multiple discovery mechanisms

- Protocol is transport independent
  - Ground-up implementation of the D-Bus wire-protocol with extensions
  - Supports Wi-Fi and Bluetooth currently
Why the D-Bus Wire Protocol?

Why reinvent the wheel?

- www.freedesktop.org/wiki/Software/dbus
- IPC mechanisms used on many Linux distributions
  - Deeply integrated with system services and session management
  - Supports RPCs as well as unicast and multicast events
- Message bus architecture
  - RPC and events implemented as messages
  - P2P came from extending bus cross device; messages flow over bigger bus
- Object oriented
  - Objects, interfaces, methods, and properties
- Language neutral
  - Bindings for C, Java, Python, Perl, etc.
DBus Compatibility

AllJoyn Functionality

Object model

DBus Compatibility

Service Advertisement & Discovery
Peer sessions
Distributed Bus Management
Device-to-device message routing
Message Expiration (TTL)
Routing rules
Signals
Remote procedure call
“Raw” Session support
Name space
Bluetooth
WiFi

Authentication & Encryption
Header Compression
Client-bus authentication
Introspection
Wire-protocol

Topology management
Client-bus authentication
Introspection
Wire-protocol
Distributed Software Bus

Conceptually peers are applications, not devices. Applications communicate with a local daemon. Daemons handle routing between devices.
Ad-hoc Bus Formation

Actual discovery mechanism is transport dependent
- On Wi-Fi a lightweight IP multicast protocol
- On Bluetooth device discovery with EIR and SDP query
Ad-hoc Bus Formation

Once connected daemons form single bus with shared namespace

- Peers can discover when other peers join or leave bus
- Peers can make RPC calls and send and receive events
- Session reference counting keeps device-to-device connections alive
- Multicast events can be sent to all peers in the session
Object Model

- AllJoyn applications expose their functionality via objects
  - These are typically organized in a hierarchy
- Objects implement interfaces (one or more)
- Interfaces are composed of members, which fall into three categories
  - Methods – classic OO object interaction
  - Signals – asynchronous event notification
    - Can be broadcast, multicast or point-to-point
  - Properties – data members
Language Bindings

- Native implementation for AllJoyn is C++
- Java binding is available for Android
- Binding for JavaScript is under development
- Considering bindings for C, C# and Python
- Object model is similar for all bindings
Design of Security Framework

Authentication and encryption is designed to be app-to-app

- The bus is not involved other than to route
- Trust relationship established between the applications
- Device pairing not required unless the transport requires it
  - In case of Bluetooth, AllJoyn does not normally trigger pairing

Security is enabled per-interface

- Authentication and key exchange initiated on demand

Security-enabled interface

- Authentication is required to make method calls
- Authentication required to receive signals
Security Model—Authentication and Encryption

Method calls to security-enabled interface are designed to be encrypted and authenticated. If call encrypted, so is reply.

Signals from a security-enabled interface are designed to be encrypted and authenticated.

Application can allow a mix of security-enabled and non-secure interfaces. Maintain own key store—can use password.
Performance Considerations
Message Efficiency

Header compression
- Designed to significantly reduce the size of message headers
- Intended for messages that are sent frequently
- Designed to allow short messages to fit into single-slot Bluetooth packets

Time to live
- Added for multi-player games and streaming where stale data is useless
- Expired messages are silently discarded
- Mainly applies to signals
- Introduces a new message header field

Multipoint sessions
- Bounds the scope of broadcast signals to session members
- Provides mechanism for deciding when radios are no longer in use
## Direct Point-to-Point Bulk Data Exchange

### Raw Sessions

- Removes overhead by providing direct access to underlying socket
- Application can read/write *any* data to the socket

### Process

1. AllJoyn sets up a regular peer-to-peer session
2. Peers call daemon to obtain the file descriptor for socket
3. Daemon returns open file descriptors to peers, closes session
4. Session is gone but sockets remain open

---

**Available in current 2.0 release**
Open Source
Why Open Source AllJoyn?

- Qualcomm Innovation Center, Inc. (QuIC)
  - Brings together dedicated group of engineers focused on OSS
  - Enables and optimizes open source software with Qualcomm technology
  - QuIC is a large contributor to WebKit and other OSS projects

- Peer-to-peer success relies on network effect
  - Must be on as many devices as possible
  - Not practical for QuIC to port to every platform
  - Allow OSS community to add platforms
    - As well as features, and fixes! All contributions are welcome!

- To further enable this chose Apache 2.0 license
  - It is a common and well accepted OSS license
  - Provides flexibility to licensees and developers
AllJoyn Open Source Project:  www.alljoyn.org

- Launched Jan. 2011
- GitHub hosts source alljoyn.github.com
- Apache 2 License
- Download
  - Documentation
  - SDK
- Contribute Code
- Participate
- Get Source
Open Source Status

- Publishing updates to the source all weekdays
- Publish a binary SDK when we do major releases
  - Released version 2.0 mid May
- Have process in place for accepting third party contributions
- Building user community
  - Lots of discussions on the forums
  - Broad range of questions and topics
Summary

Easier P2P Development

- OS and platform independent
- Multiple language bindings
- Multiple transports

Open Source Sponsored by QuIC

- Launched January 2011
- SDK download and forums at www.alljoyn.org
- Source code available on GitHub at www.github.com/alljoyn

Download SDK  Download Source Code  Contribute
Questions?
Disclaimer

Copyright © 2011 Qualcomm Innovation Center, Inc. All rights reserved. AllJoyn is a trademark of Qualcomm Innovation Center, Inc. Other product and brand names may be trademarks or registered trademarks of their respective owners.