AllJoyn™ Overview and Integration Tips & Tricks

Brian Spencer
Staff Engineer
Qualcomm Innovation Center, Inc.
August 30, 2012
Agenda

- What is AllJoyn?
- What About Other Peer-to-peer Solutions?
- AllJoyn Fundamentals
- Add AllJoyn to an Android Application
- Simple Client/Service Sample Walkthrough
- How to build and run AllJoyn at OS layer
- Q&A
What is AllJoyn?
What Is AllJoyn?
Open Source Application Development Framework to Enable Ad Hoc, Proximity-based, Peer-to-peer Networking

AllJoyn brings proximity awareness to mobile apps, unleashing a whole new set of user experiences to smartphones, tablets, PCs, TVs and more
AllJoyn is a Software Framework

- Open Sourced (Apache 2.0 License)
- Enables Developers to Easily Add P2P Experiences to Their Apps
- Application Layer Discovery (What services are running on nearby devices that are reachable)
- Application Layer Security (What information can a service access on your phone, what’s off limits)
- Interoperate Across Different OS and Bearers (Developer does not need to know anything about Bluetooth, Wi-Fi, Android, Windows, etc.)

Applications
- JamJoyn™
- Media
- Multiplayer Gaming
- Social
- Chat
- Third-Party or OEM

AllJoyn Development Framework
- C++
- Java
- JavaScript
- Objective-C
- CLR
- Discovery
- Security
- Network Management
- Pairing
- Cross-Platform
- Wireless-Optimized

Chipset
- Wi-Fi Direct
- Wi-Fi
- BT

Devices
- Tablets
- Mobiles
- Televisions
- Laptop/PC

Open Source. Open Possibilities.
What About Other P2P Solutions?
What About Existing Protocols?

- **AllJoyn**
  - Multiple wireless transports
  - Application Centric
  - Device Centric
  - Network management
  - App development framework
  - Media streaming & security
  - Control plane
  - Discovery
  - IP Transport

- **DLNA®**
- **UPnP**
- **Bonjour**

**Other peer-to-peer platforms focus on their own ecosystem**

**Could be standards that are slow to change and fixed in design**

**AllJoyn is a complete package that works across different operating systems and programming languages to provide a complete solution**
What About Existing Protocols?

NFC is for small data loads and devices must be touching.

AllJoyn can be adopted to use NFC as a transport:
- Could be a great discovery transport with communication occurring over WiFi or BT.

<table>
<thead>
<tr>
<th>Feature</th>
<th>AllJoyn</th>
<th>DLNA®</th>
<th>UPnP</th>
<th>Bonjour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple wireless transports</td>
<td>[Blue]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application Centric</td>
<td>[Blue]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device Centric</td>
<td></td>
<td>[Yellow]</td>
<td></td>
<td>[Purple]</td>
</tr>
<tr>
<td>Network management</td>
<td>[Blue]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>App development framework</td>
<td>[Blue]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Media streaming &amp; security</td>
<td></td>
<td>[Yellow]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control plane</td>
<td>[Blue]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discovery</td>
<td>[Blue]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP Transport</td>
<td>[Blue]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PAGE 8  Open Source. Open Possibilities.
What About Existing Protocols?

- **WiFi Direct** is much like BT with device pairing
  - Focus is on establishing IP networks

- **AllJoyn** avoids the complications of pairing devices
  - Provides higher level API's that work across different wireless protocols

<table>
<thead>
<tr>
<th>Feature</th>
<th>AllJoyn</th>
<th>DLNA®</th>
<th>UPnP</th>
<th>Bonjour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple wireless transports</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application Centric</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device Centric</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>App development framework</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Media streaming &amp; security</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control plane</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discovery</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP Transport</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## What Operating Systems and Languages?

### TODAY
- **Android**
  - 2.1 (Éclair) and newer
- **Windows® 7** and XP
- **Linux**

### BETA RELEASE
- **Windows® RT**

### IN DEVELOPMENT
- **iOS**

### IN DEVELOPMENT/DEMONSTRATED
- **JavaScript**
- **C#** (General)

### LANGUAGE BINDINGS
- **C++**
  - (native implementation)
- **Java**
- **C**
- **C#** (Unity)

AllJoyn is open source and available [http://www.alljoyn.org](http://www.alljoyn.org)
AllJoyn Fundamentals
AllJoyn Fundamentals

AllJoyn is a distributed software bus
- Each device runs a bus daemon
- Applications communicate directly only with the 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
- Supports Wi-Fi and Bluetooth currently
- Working on Wi-Fi Direct
Bus Attachments, Objects, Proxy Objects

An application needs a Bus Attachment to communicate with the bus

- Bus Attachments provide a root (/) for the object hierarchy

Bus Objects implement interfaces

- Bus Objects path names look like file paths, e.g. /org/AllJoyn/Games/chess
- Bus Object have methods than can be called remotely
- Bus Objects can emit signals

Proxy Bus Objects are local representations of remote Bus Objects.

- Applications use proxy bus objects to make method calls to remote objects
Connect to the AllJoyn Bus

```java
mBus = new BusAttachment(getClass().getName(), BusAttachment.RemoteMessage.Receive);
mBus.useOSLogging(true);
mBus.setDebugLevel("ALLJOYN_JAVA", 7);
mBus.registerBusListener(new LocalBusListener());

status = mBus.connect();
if(Status.OK != status) /*ERROR*/
```

This object represents the connection to the bus (daemon)

Connect the attachment to the bus

© 2012 QUALCOMM Incorporated. All rights reserved.

* Code snippets licensed under the Apache License, version 2.0 http://www.apache.org/licenses/LICENSE-2.0
Register Bus Objects

/* Define an interface that will be your AllJoyn interface for P2P communication */
@BusInterface (name = "org.alljoyn.bus.samples.training")
public interface AllJoynTrainingInterface {
    @BusMethod(signature = "s")
    public void TraningMethod(String arg) throws BusException;
}

-------------
class TrainingService implements AllJoynTrainingInterface, BusObject {
    public void TraningMethod(String arg) { /* some code */ }
}

-------------
theService = new TrainingService();
Status status = mBus.registerBusObject(theService, "/TrainingService");
if(Status.OK != status) {"ERROR"}

* Code snippets licensed under the Apache License, version 2.0 http://www.apache.org/licenses/LICENSE-2.0

© 2012 QUALCOMM Incorporated. All rights reserved.
Register Signal Handler

/* Define an interface that will be your AllJoyn interface for P2P communication */
@BusInterface (name = "org.alljoyn.bus.samples.training")
public interface AllJoynTrainingInterface {
  @BusSignal(signature = "s")
  public void SignalMethod(String arg) throws BusException;
}

/* in the application register the class that implements the SignalMethod handler*/
status = mBus.registerSignalHandlers(this);
if(Status.OK != status) { /*ERROR */}

/* Here is the handler */
@BusSignalHandler(iface = "com.alljoyn.bus.samples.training", signal = "SignalMethod")
public void SignalMethod(String arg) { /* some code */ }

Define the Interface that represents the methods of your P2P application
Let the Bus know what class contains the handler for signals
This method executes when a signal is sent out assuming device is connected on same session
Advertise Well-Known Name

Status status = mBus.advertiseName("com.alljoyn.org.samples.training", SessionOpts.TRANSPORT_ANY);
if(Status.OK != status) {
    /*ERROR – Failed to advertise name*/
    status = mBus.releaseName("com.alljoyn.org.samples.training");
}

Register with the bus that we are going to be aliased the com.alljoyn.org.samples.training class

* Code snippets licensed under the Apache License, version 2.0 http://www.apache.org/licenses/LICENSE-2.0
© 2012 QUALCOMM Incorporated. All rights reserved.
Discover Well-Known Names

class LocalBusListener extends BusListener {
    public void foundAdvertisedName(String name, short transport, String namePrefix) {
    }
    public void lostAdvertisedName(String name, short transport, String namePrefix) {
    }
    public void nameOwnerChanged(String busName, String previousOwner, String newOwner) {
    }
}

Listener class for discovery events. This is where we are informed of other services

Look for a com.alljoyn.org.samples.training service

Status status = mBus.findAdvertisedName("com.alljoyn.org.samples.training");
if(Status.OK != status) {

} /*ERROR */
Create a Session

```java
class MySessionPortListener extends SessionPortListener {
    public boolean acceptSessionJoiner(short sessionPort, String joiner, SessionOpts sessionOpts) {
        return true;
    }
    public void sessionJoined(short sessionPort, final int sessionId, String joiner) {
        mBus.setSessionListener(sessionId, new MySessionListener());
    }
}
```

Listener class for session events. Lets us accept sessions and informs when users join.

```java
Mutable.ShortValue agreedUponPort = new Mutable.ShortValue(55); /* value can be 1 to 32767 (max short) */
SessionOpts sessionOpts = new SessionOpts();
    sessionOpts.traffic = SessionOpts.TRAFFIC_MESSAGES;
    sessionOpts.isMultipoint = true;
    sessionOpts.proximity = SessionOpts.PROXIMITY_ANY;
    sessionOpts.transports = SessionOpts.TRANSPORT_ANY;
Status status = mBus.bindSessionPort(agreedUponPort, sessionOpts, new MySessionPortListener());
if (Status.OK != status) { /*ERROR – Could not create a session*/ }
```

We create the session based on the Session Options we pass. We can specify the transport interface here for the supported types.

### Application

**Bus Attachment**

* Code snippets licensed under the Apache License, version 2.0 http://www.apache.org/licenses/LICENSE-2.0

© 2012 QUALCOMM Incorporated. All rights reserved.
Join a Session

```java
short agreedUponPort = 55; /* value can be 1 to 32767 (max short) */
SessionOpts sessionOpts = new SessionOpts();
Mutable.IntegerValue sessionId = new Mutable.IntegerValue();
Status status = mBus.joinSession("com.alljoyn.org.sample.training", agreedUponPort, sessionId,
   sessionOpts, new MySessionListener());

if(Status.OK == status) {
   /* NOW CONNECTED */
} else if(status == Status.ALLJOYN_JOINSESSION_REPLY_ALREADY_JOINED) {
   /* ALREADY JOINED */
} else {
   /* ERROR */
}
```

We have already found the name so we now join the session

```
class MySessionListener extends SessionListener {
   public void sessionLost(int sessionId) {}
   public void sessionMemberAdded(int sessionId, String uniqueName) {}
   public void sessionMemberRemoved(int sessionId, String uniqueName) {}
}
```
Add AllJoyn to an Android Application
Download Packages

Download Android SDK

Download & Setup Eclipse

Download AllJoyn SDK
  - http://alljoyn.org/docs-and-downloads
Steps to Add AllJoyn to Existing Application

Download complete documentation here:
- [https://www.alljoyn.org/content/guide-alljoyn-development-using-java-sdk](https://www.alljoyn.org/content/guide-alljoyn-development-using-java-sdk)

First: Import AllJoyn libraries
- Create libs folder that contains:
  - alljoyn.jar
  - armeabi/liballjoyn_java.so

Second: Modify manifest to include permissions

Third: Add AllJoyn code
Simple Client/Server Sample Code Walkthrough — C++

Found in SDK
Simple Client/Service Sample Walkthrough — C++

C++ Simple sample found in the SDK:
- alljoyn-sdk-2-3-6-android-rel\samples\simple\client
- alljoyn-sdk-2-3-6-android-rel\samples\simple\service
How to build and run AllJoyn at OS layer
Linux build instructions

Complete documentation here:

Download & Setup Environment
- These tools are need at a minimum:
  - Python
  - SCons
  - Git
  - Repo
  - Java

Download AllJoyn source code
- http://alljoyn.github.com/download-source.html
Linux build instructions

Compile for Linux:

- `scons CPU=x86 VARIANT=release`
- `scons CPU=x86-64 VARIANT=release`

Compile for Android: *

- `scons OS=android CPU=arm ANDROID_NDK=/local/mnt/workspace/brian/android-ndk-r6b
  ANDROID_SRC=/local/mnt/workspace/ICS ANDROID_TARGET=generic VARIANT=release`

*Android Source code required to build for Android
Live demonstration
Questions?
Thank you!