ALLSEEN ALLIANCE

Introduction to AllJoyn™ Thin Library

June 30, 2014

Contents

Figures

1 Preface

This document provides a high-level introduction to the AllJoyn™ Thin Core Library (AJTCL). It is a companion to the overall system architecture document, *Introduction to theAllJoyn Framework*, and concentrates on the aspects of the system that relate directly to introducing embedded devices into the overall AllJoyn system architecture.

1.1 Purpose

This document describes how devices constrained in energy, computing power, and memory (cf. embedded devices) may participate in an AllJoyn distributed system. Additionally, this document covers AllJoyn abstractions relating specifically to embedded devices and describes, at a high level, how the various components of AllJoyn-based systems can work together with embedded devices to provide an overall environment for proximity-based peer-to-peer computing.

1.2 Scope

This document will be appropriate for anyone interested in adding embedded devices to peer-to-peer networks, including application developers, system software developers, network specialists, device manufacturers, and systems managers. We assume a basic understanding of embedded systems, and we assume that the reader understands the AllJoyn system to the level described in *Introduction to the AllJoyn Framework*.

2 Overview

AllJoyn is an open source software system that provides an environment for distributed applications running across different device classes, with an emphasis on mobility, security, and dynamic configuration. AllJoyn is “platform-neutral,” meaning it was designed to be as independent as possible of the specifics of the operating system, hardware, and software of the device on which it is running.

Components of the AllJoyn Standard Core Library (AJSCL) are designed to run on Microsoft Windows, Linux, Android, iOS, OS X, OpenWRT, and the Unity plug-in for internet browsers. A common characteristic of all of these software systems is that they run on general-purpose computers. General purpose computers usually have significant amounts of memory, available energy, and computing power, along with significant operating systems that support multiple processes and multiple threads with multiple standard language environments.

An embedded system, on the other hand, is one designed to provide *specific* functionality running on a microcontroller embedded within a larger device. Since an embedded system need only perform a specific function or a small number of functions, engineers are free to optimize them to reduce the size and cost of the product, often by limiting memory size, processor speed, available power, peripherals, user interfaces, or all of the above. AllJoyn Thin Core Library (AJTCL) is designed to bring the benefits of the AllJoyn distributed programming environment to embedded systems.

Since the operating environment in which an AJTCL will run may be very constrained, an AllJoyn component running on such systems must live within those constraints. This means, specifically, that we do not have the luxury of bundling in an AllJoyn router (which requires multi-threading), having many network connections, and using relatively large amounts of RAM and ROM. We do not have the luxury of running an object-oriented programming environment that includes alternate language bindings. Because of this, the AJTCL consists only of what amounts to a bus attachment (see *Introduction to the AllJoyn Framework*) written solely in the C language. The data structures corresponding to interfaces, methods, signals, properties, and bus objects are highly optimized for space, and the developer APIs are, therefore, quite different.

Although the APIs may be different, all of the major conceptual blocks found in AJSCL can be found in AJTCL systems; they just take on a more compact form or are actually run remotely on another, more capable machine.

Note When we mention the AllJoyn Standard Library (AJSCL), we explicitly refer to the versions of these components that run on general purpose computers.

3 Conceptual Model

As implied in the previous section, most high-level abstractions used in AJTCL are identical to those in the AJSCL system. The document, *Introduction to the AllJoyn Framework* has a section titled Conceptual Overview that walks you through these abstractions. In the Conceptual Overview section, we assume that the reader is familiar with the abstractions introduced in that document, so we will only touch on the differences that are required to understand the AJTCL architecture.

3.1 AllJoyn Thin Core Library is still AllJoyn

It is important to understand that AJTCL is part of the AllJoyn framework. A Thin Core Library is completely interoperable with AJSCL. Since the AllJoyn network wire protocol is completely implemented on both types of such a system, AJSCL can be completely unaware of the fact that they are talking to Thin Core Libraries, and vice versa.

Recall from *Introduction to the AllJoyn Framework* that the basic structure of an AllJoyn distributed bus consists of multiple bus segments residing on physically separate host computers as shown in *Figure 1*.

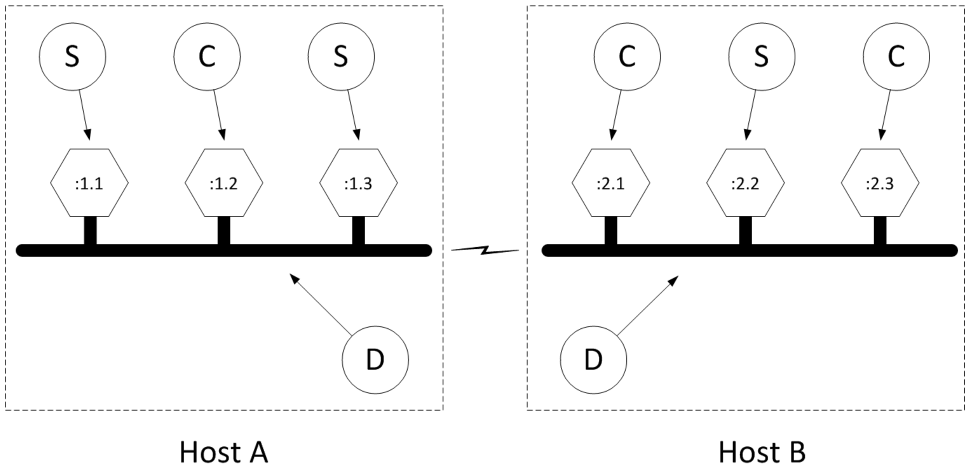

Figure 1: An AllJoyn distributed bus

Recall that each bus segment is located on a given host computer, as illustrated by the dotted squares labeled Host A and Host B in the figure. Each bus segment is implemented by an AllJoyn router (shown as the bubbles labeled D in the figure). There may be several bus attachments on a host, each connected to the local daemon (illustrated by hexagons). These hexagons are refined to be services (S) or clients (C).

Since the host computer running AJTCL typically does not have the resources to run a router, the AllJoyn architecture changes things such that to connect to the distributed bus the Thin Core Library borrows an AllJoyn router running on another host computer. This arrangement is shown in *Figure 2*. Notice that Embedded System A and Embedded System B are not the same devices as Host B, which is running the router that manages the distributed bus segment on which the embedded devices reside. The connection between the embedded systems running AJTCL and the router hosting the bus segment is made through Transmission Control Protocol (TCP).

The network traffic flowing between the embedded systems and the routers are AllJoyn messages implementing bus methods, bus signals, and properties flowing over their respective sessions, as described in *Introduction to the AllJoyn Framework*.

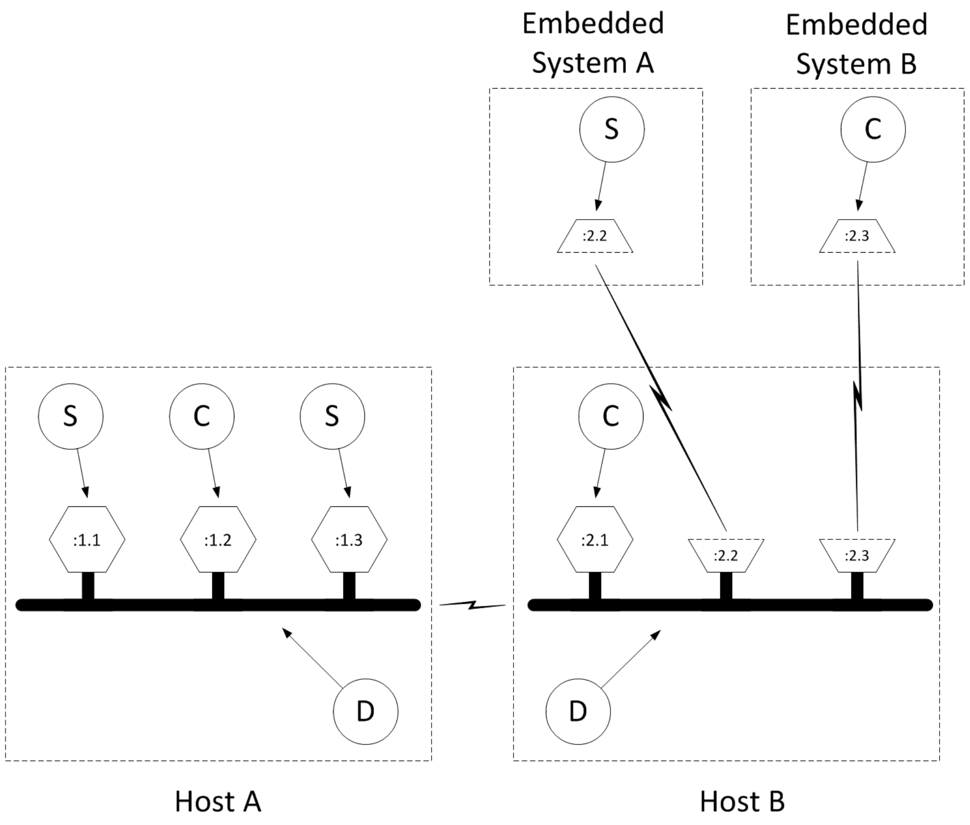

Figure 2: An AllJoyn distributed bus with thin core libraries

It is sometimes desirable to allow AJTCL devices to connect to and borrow any old router found in the proximity. We call these untrusted relationships (from the router perspective). It is also sometimes desirable to allow only particular AJTCL devices to connect to specific routers. We call these trusted relationships (again, from the router perspective).

These relationships are established using a discovery and connection process that is conceptually similar to the discovery and connection process of clients and services. An AllJoyn router conveys its willingness to host a given collection of AJTCL devices by advertising a well-known name. This advertisement may be driven either by router configuration or by an advertisement specifically made by an AllJoyn component. When a connection attempt is made to any router as a result of a discovery event, a router expecting trusted relationships may choose to challenge a particular Thin Core Library (or impersonator of a Thin Core Library) to produce a credential. In the case of an untrusted relationship, the router may choose to simply allow any connection attempt. In the case of an untrusted connection, the involved router will not allow the Thin Library to perform any operations that will cause sessions to be established with components off the local device (and which, therefore, correspond to a “service that costs you money”).

As implied above, the connection process for an AJTL device is split into three phases:

- Discovery phase
- Connection phase
- Authentication phase

The discovery phase works just like service advertisement and discovery as described in *Introduction to the AllJoyn Framework*, with two exceptions. The first exception is that advertisements for the purpose of AJTL discovery are typically “quiet” advertisements. This simply means that the advertisements are not sent gratuitously by the router.

The second exception is that responses to quiet advertisements are sent quietly – we call these quiet responses. This means that the responses are unicast back to the requester instead of being multicast as they are in "active" advertisements. The primarily reason for this change is to allow embedded devices that do not fully implement multicast reception to participate in AllJoyn distributed systems.

3.2 What is an AllJoyn Thin Core Library device?

One typically thinks of an AJTCL device as conceptually similar to a Sensor Node (SN) in a Wireless Sensor Network (WSN). Sensor nodes are typically sensors/actuators that are small in size and constrained in energy, computing power, memory, or other resources. They are able to sense their surroundings, communicate events to the outside world, and possibly take actions based on internal processing or as a result of external events. This is a very broad definition, and a small sampling of the sort of devices that might fit into such a definition could be:

- Light switches
- Thermostats
- Air conditioners
- Vent dampers
- Smoke detectors
- Motion detectors
- Humidity detectors
- Microphones
- Speakers
- Earphones

- Doors
- Doorbells
- Ovens
- Refrigerators
- Toasters

There is a large amount of literature available that discusses wireless sensor networks (WSNs). AllJoyn systems are distinguished from such networks in that WSNs typically use self-organizing multi-hop ad hoc wireless networks where security is not a major concern; whereas the AllJoyn framework will most likely run on infrastructure-mode Wi-Fi networks to which a given device must be associated and authenticated. In order to accomplish the secure admission to a Wi-Fi network, AJTCL uses a process called “onboarding.” The Onboarding service framework allows a Thin Core Library device, which presumably has no friendly user interface, to learn enough information about its destination network to accomplish the admission and authentication processes required to join that network. The Onboarding service framework is defined in detail in a dedicated document.

In its role as a kind of sensor node, an AJTCL device typically implements a service in the AllJoyn sense. It senses its surroundings using attached hardware and communicates events to the outside world through AllJoyn signals. It can take actions as a result of external events, either by listening for signals from other devices or by responding to Remote Method Invocations from AllJoyn clients, as discussed in *Introduction to the AllJoyn Framework*.

4 Thin Core Library Architecture

Since the AllJoyn Thin Core Library (AJTCL) must run in devices that are constrained in energy, processing power, and memory, such devices do not have the luxury of using the same architecture as a general-purpose computer system running AllJoyn Standard Core Library (AJSCL).

The layered architecture of an AJSL or service process is reproduced in *Figure 3*. Refer to *Introduction to the AllJoyn Framework* for a more detailed discussion of these layers. The important observation to make at this point is that each AllJoyn client or service reproduces this layering in every process representing an AllJoyn application.

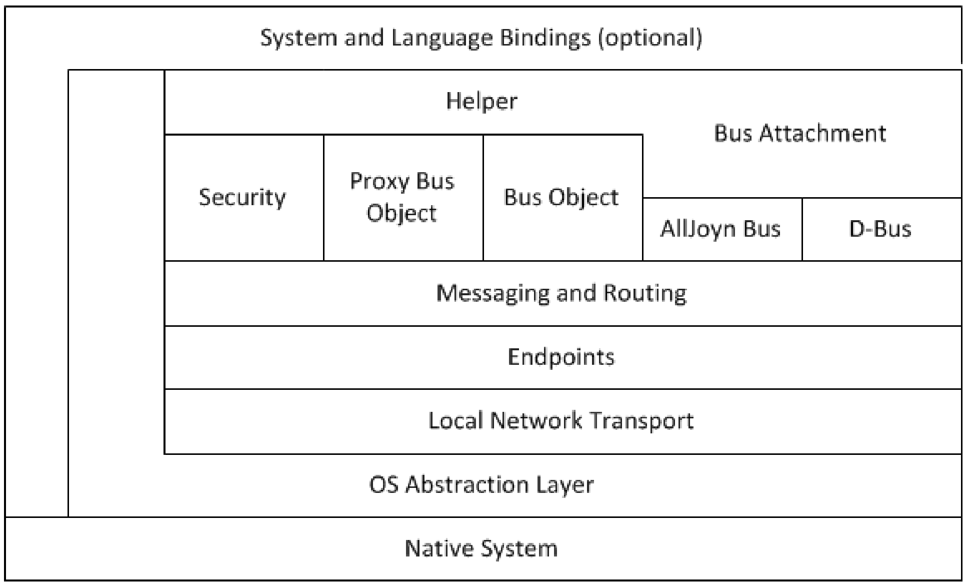

Figure 3: AJSCL layering

Every AJSCL-enabled host needs to have at least one AllJoyn router. This router may reside in its own process in the standalone router case, or it may be co-located with an application in the bundled router case. The layered architecture of an AJSCL router is reproduced in *Figure 4*.

Notice that the router adds additional support for routing messages between router, along with the capacity to use a multiple network transport mechanisms such as Wi-Fi Direct. This is a significant amount of functionality and comes at a considerable cost in computing power, energy, and memory.

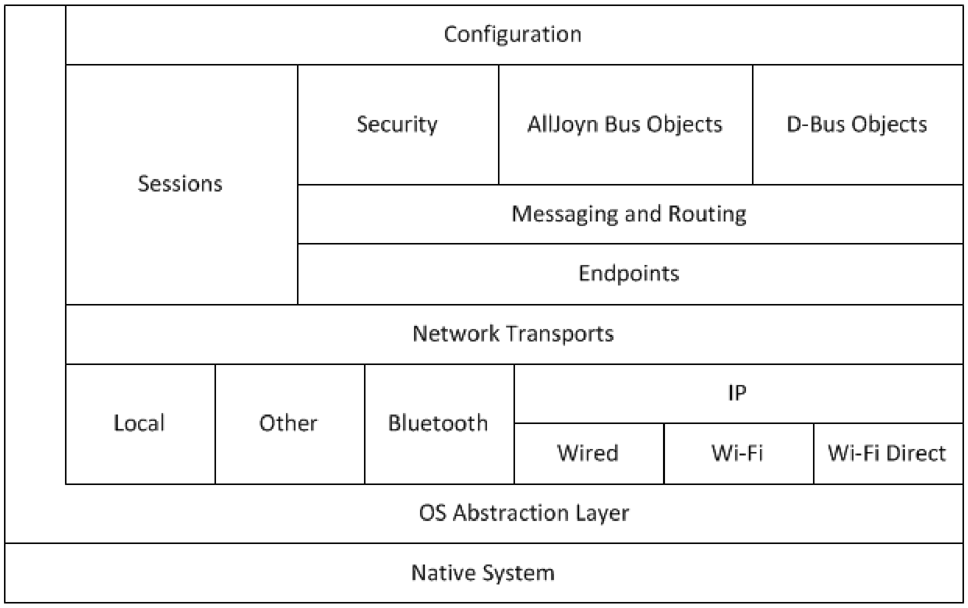

Figure 4: AJSCL router layering

Clearly, it is not possible to run this significant amount of code in a constrained embedded system, so AJTCL minimizes the amount of this code that is required to exist on a given device. It does this by constraining the basic environment to a minimal C-only run-time, and by borrowing other devices to perform the router role for it. In contrast to AJSCL, AJTCL, as shown in *Figure 5*, does away with much of the overhead present in the AJSL system. AJTL exposes only the minimum required API to the bus attachment and exposes the AllJoyn messaging interface directly instead of providing helper functions.

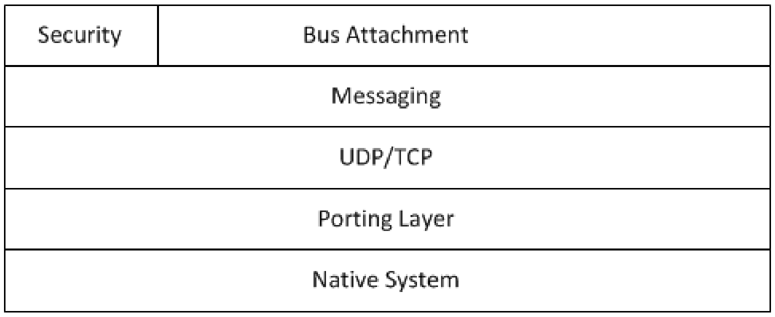

Figure 5: AJTCL layering

Instead of providing an abstract transport mechanism, the messaging layer uses User Datagram Protocol (UDP) and TCP directly. There is a very thin porting layer to abstract a few needed native system functions, and the entire package is written in C, with an eye toward minimizing code size. Because of these optimizations, an AJTCL system can run in as little as 25 Kbytes of memory, whereas a bundled router and C++ client or service combination may require ten times that amount, and a Java language version may require as much as 40 times that footprint.

5 Tying it All Together

In order to make this discussion somewhat more concrete, two example distributed systems are presented here. The first example is a minimal system in which a single AllJoyn application running on a smartphone talks to a single AJTCL device. This illustrates the trusted router relationship as described above. The second example is a more complicated system with a router running on a wireless router.

Note Typically, this situation would be a router running OpenWRT that hosts a preinstalled AllJoyn router. This router accepts untrusted connections from Thin Core Libraries that have been onboarded to the Wi-Fi network.

A small number of AJTCL devices connect to the router and act as the sensor nodes for an AllJoyn-based wireless sensor network, and a general purpose computer performs the data fusion function.

Note In Wireless Sensor Networks, data fusion is a term that refers to a process where some distinguished node collects results from some number of sensor nodes and integrates, or “fuses,” its results with those of the other sensor nodes and makes some decision on an action to take as a result of this data.

5.1 A minimalist Thin Core Library system

A minimal example of a system using a AJTCL consists of a single host running AJSCL and a Thin Core Library device. AJSCL provides the AllJoyn router which the Thin Core Library will attach to, and also provides a platform for running an application that uses the Thin Core Library. As mentioned above, the Thin Core Library typically acts as a kind of sensor node, and sends data to an application running on the host. The application typically processes the data in some way and issues commands to the sensor to manipulate its environment.

For a plausible but simple system, consider a wall thermostat that controls a furnace, and a control application running on an Android device. The Android device will run AJSCL, and the wall thermostat will run the AJTCL. Such a system is illustrated in *Figure 6*.

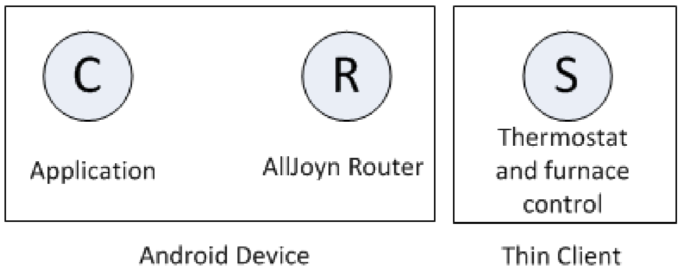

Figure 6: Minimalist example system

In this example, a requirement is that the wall thermostat only be controllable by a corresponding thermostat controller application in the Android device.

Since a requirement of the example is that the thermostat be controllable only by the Android device, it is probably also a requirement that the thermostat associate itself with only a router associated with the application. This implies that the Android application should be bundled with an AllJoyn router and only this particular combination of bundled router and application should advertise itself as a router for the Thin Core Library to use. This kind of arrangement leads to a trusted relationship between AJTCL and the router/application pair.

The application then asks its bundled router to quietly advertise a well-known name that is known to AJTCL (for example, com.company.BusNode<guid>). The router is then primed to respond to discovery requests for that name in the form of quiet (unicast) responses. When the Thin Core Library comes up, it will perform discovery on the associated network prefix (com.company.BusNode) as shown in *Figure 7*.

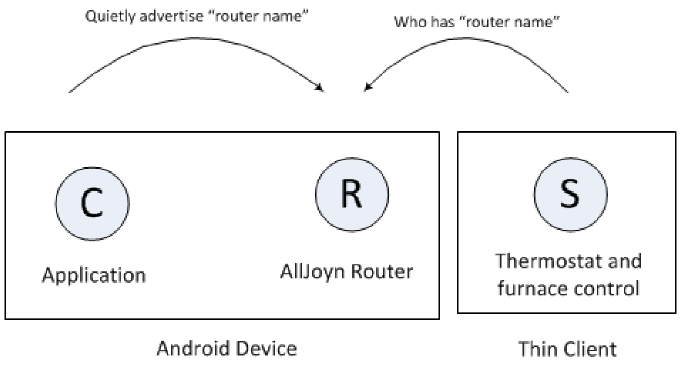

Figure 7: Thin Core Library router discovery

When the router receives the explicit inquiry about a name it is quietly advertising, it will respond with an indication that the requested name "is at" the particular router. AJTCL will then attempt to connect to the responding router as shown in *Figure 8*.

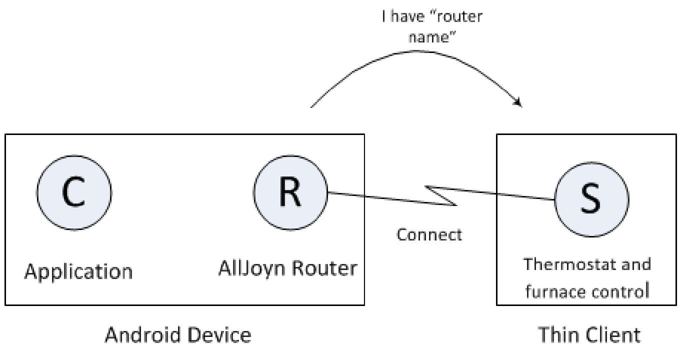

Figure 8: Thin Core Library connection attempt

At this point, a logical AllJoyn bus has been formed, in which both the application and Thin Core Library service are associated with the bundled router running on the Android device. Representing the system using the bubble diagrams used in *Introduction to the AllJoyn Framework*, the arrangement appears as if the AllJoyn router has a connected service and client, as shown in *Figure 9*.

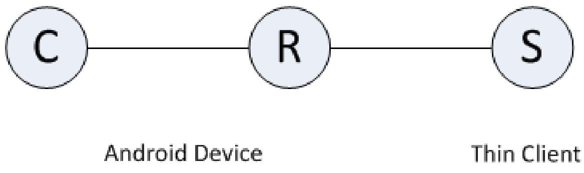

Figure 9: Bubble diagram of example Thin Core Library system

At this time, the AJTCL is connected to the router bundled with the application, but neither the application nor the Thin Core Library knows of each other's existence. Typically at this time, AJTCL would request a well-known bus name and instantiate a service in the AllJoyn sense. The Thin Core Library would create a session port and advertise a well-known name as described in *Introduction to the AllJoyn Framework* using the Thin Core Library APIs. This well-known name would typically be different than the well-known name that the bundled router advertises; it corresponds to the client/service relationship between the Thin Core Library and the application, rather than the relationship between the router and the Thin Core Library. The application running on the Android device would then perform service discovery for that name as shown in *Figure 10*.

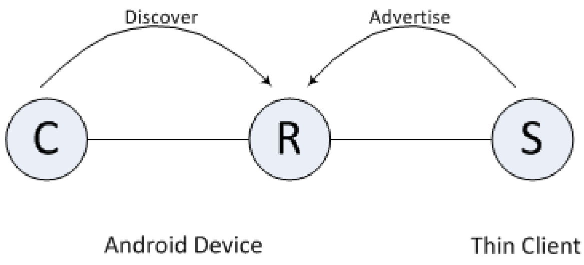

Figure 10: Service discovery with the Thin Core Library

When service running on AJTCL is discovered by the client running on the Android device, the client may join the session created by the service.

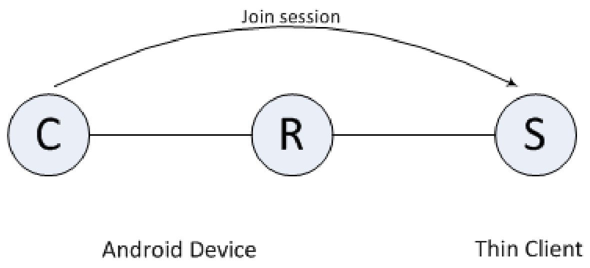

Figure 11: Android device joins session with service on the Thin Core Library

At this point, the application running on the Android device may access the AJTCL service, as it would any AllJoyn service. It may choose to be notified of signals emitted by the service – in this case, perhaps periodic signals consisting of the current temperature. The application may choose to present a user interface that allows a user to enter a desired temperature and then send that temperature to AJTCL using AllJoyn remote method invocation as described in *Introduction to the AllJoyn Framework*. Upon receiving a Method Call, the service running in AJTCL could relay the request to the furnace to set the desired temperature.

The API used on the Thin Core Library side is considerably different from that used in AJSCL or a service; however, since the wire protocol is identical in both cases, the flavor of a component on the other side of the connection (AJSCL or AJTCL) is not visible. At this point, AllJoyn is AllJoyn and the bubble diagrams, including AJTCLs, are indistinguishable for all intents and purposes from those bubble diagrams shown in the *Introduction to the AllJoyn Framework* document.

5.2 A Thin Core Library-based wireless sensor network

This example composes a very basic home management system. The wireless access point is assumed to be an OpenWRT router that hosts a preinstalled AllJoyn router that allows for untrusted Thin Core Library connections. This will allow all AJTCLs participating in the system to connect to the router daemon. Thin Core Library devices in this network could be temperature sensors, motion detectors, light switch actuators, water heater thermostats, furnace or air conditioning system temperature controllers.

As described above, the data fusion function for the example network is performed by an application running on a general purpose computer system with an integrated display. It is not required that there be a dedicated general-purpose computer in the network – data fusion can be accomplished in a distributed fashion; however, having this component present in the network allows us to illustrate how AJSCL and Thin Core Library devices can interoperate. The “fuser” display could be mounted on a wall in the home or it could simply be the display of a PC located somewhere in the home. This display can, for example, provide user interface elements corresponding to thermometers and thermostats for individual rooms; or virtual light switches, or motion detectors. The actual data fusion function algorithms would determine when to turn lights, home heat, or air conditioner on or off, or when to turn the water heater temperature up or down in the most efficient way.

The first component considered is the OpenWRT router and is illustrated in *Figure 12*. The router hosts a standalone AllJoyn router daemon that we illustrate as the bold horizontal line that represents a segment of an AllJoyn distributed software bus.

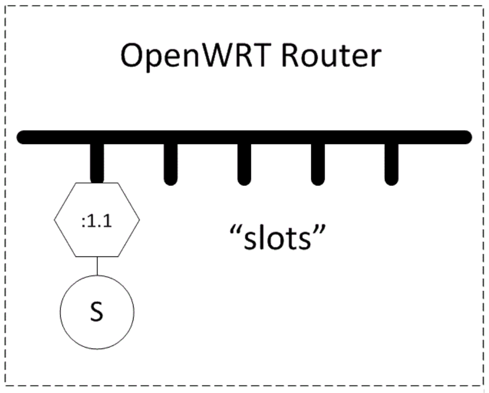

Figure 12: An OpenWRT router hosting a standalone AllJoyn router daemon

There may be an AllJoyn service residing on the router's bus segment that provides a way to configure the router and the preinstalled router using the AllJoyn framework itself. In addition, there are a number of empty slots that represent untrusted connections to AJTCLs. Since this is a generic AllJoyn router, the corresponding software bus may be extended to other bus segments to form a distributed bus as shown in *Figure 1*.

As described in the previous section, AJTCL devices will perform discovery to search for a router to which they can connect. Since an untrusted relationship is described here, the AllJoyn router running on the OpenWRT router will be configured to quietly advertise a generic name, perhaps org.alljoyn.BusNode, implicitly indicating that the router is a node on an AllJoyn distributed bus willing to host Thin Libraries.

AJTLs representing the sensor nodes in the distributed network are brought onto the wireless network through the onboarding process. During this process, they may be assigned so-called friendly names which give them meaning in the context of the home. For example, one light bulb actuator (on-off-dim switch) might be given the name "Kitchen" and another the name "Living Room." The corresponding Thin Core Library nodes begin discovery of their assigned router (perhaps org.alljoyn.BusNode) and will then make connection attempts. Since the slots in the preinstalled router running in the OpenWRT router are presumably untrusted, the Thin Core Library connections are accepted on the network as illustrated in *Figure 13*.

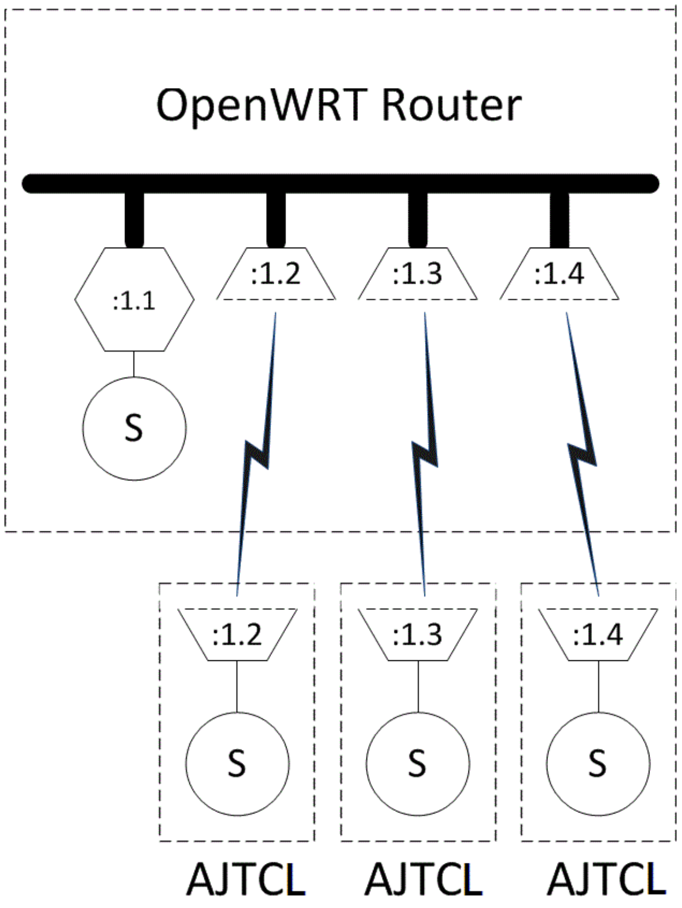

Figure 13: AJTCL nodes connected to the OpenWRT AllJoyn router

Once the Thin Core Library Apps are connected to the bus segment implemented in the OpenWRT router, they begin to advertise their corresponding services. Presumably, there is also a home control system onboarded to the wireless network provided by the router. This device will be doing service discovery and looking for the service provided by the Thin Core Libraries in the system, as shown in *Figure 14*.

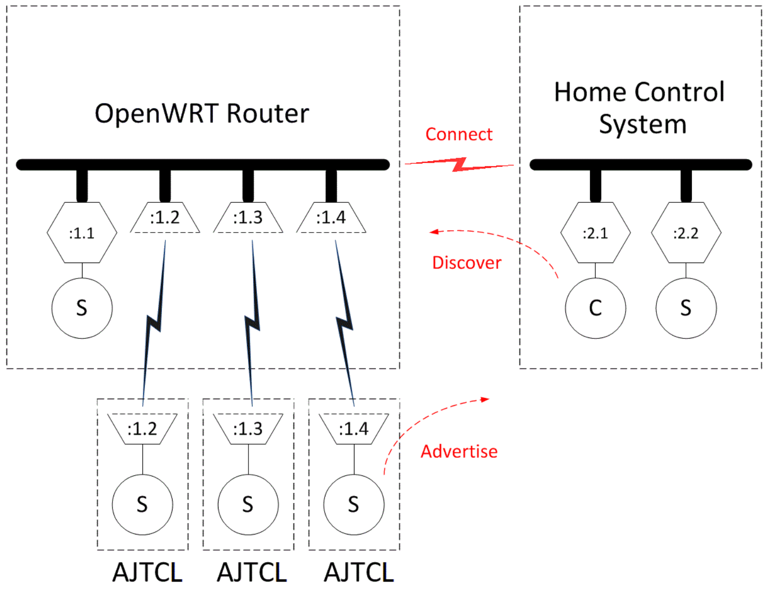

Figure 14: OpenWRT router, Thin Core Libraries, and home control system

Once the home control system has discovered the service advertisements of one of AJTCLs, it will attempt to join a session with the discovered Thin Core Library as discussed in *Introduction to the AllJoyn Framework*. This will result in the bus segments implemented on the router and the home control system merging into a single virtual distributed bus.

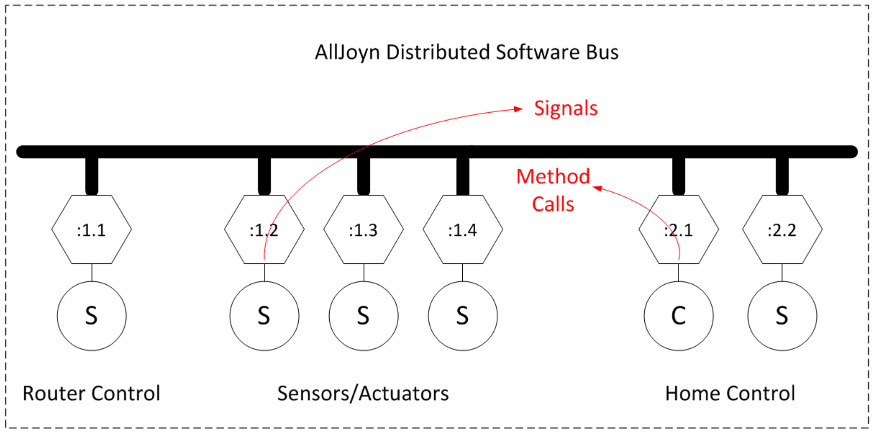

Figure 15: AllJoyn distributed software bus

When the merged bus is fully formed, the devices attached to the bus behave as generic AllJoyn clients or services. The fact that AllJoyn Thin Core Library sensors and actuators are actually embedded devices connected to an AllJoyn router over TCP is not exposed to other components on the distributed bus. The fact that the home control system is perhaps written in Java and running on a general purpose computer running Android is not exposed to other components on the distributed bus. The clients and services simply make and implement remote method calls and emit and receive signals.

The algorithms running in the data fusion node can now be understood clearly. For example, one important AllJoyn signal sent over the distributed bus might be something corresponding to `CARBON-MONOXIDE-DETECTED`. This signal would be received by the home control system (the data fuser) and it might react by sending a remote method call to one of the actuator nodes telling it to `TURN-FAN_ON`, it might send a remote method call to another actuator node telling it to `SOUND-ALARM`, and it might also send an SMS message to the homeowners letting them know that excess carbon monoxide has been found in the home.

More mundane functions of the home control system might be to make a remote method call to the furnace to reduce the temperature of the home if nobody is present (as reported by motion detectors and a daily schedule). The home control unit may send a message to the water heater telling it to reduce the temperature of the water during the work day or in the middle of the night, but may make a method call to turn the water temperature up in the middle of the night so that the dishwasher can be run at a time corresponding to the least expensive cost of electricity.

All of the signals that the home control system reacts to and the method calls made are completely independent of the type and location of the source and sink devices.

6 Summary

AllJoyn is a comprehensive system designed to provide a framework for deploying distributed applications on heterogeneous systems. The AJTCL enables embedded devices to participate in an AllJoyn distributed software bus and present themselves to the rest of the system in such a way as to abstract out the details that usually plague developers in such heterogeneous systems. This approach lets application developers focus on the content of their applications without requiring a large amount of low-level embedded system or networking experience.

The AllJoyn system is designed to work together as a whole and does not suffer from inherent impedance mismatches that might be seen in ad-hoc systems built from various pieces. We believe that the AllJoyn system can make development and deployment of distributed applications that include embedded system components significantly simpler than those developed on other platforms.

7 Learn More

To learn more about how to integrate AllJoyn in your development efforts, access the documentation and downloads available on the AllSeen Alliance web site: (www.allseenalliance.org).

- Introductory guides - Describe AllJoyn technologies and concepts.
- Development guides - Provide guidelines to setting up the build environment and provide solutions to specific programming problems, including code snippets and explanations.
- API references - Provide details for working with the AllJoyn source code and writing applications in each supported programming language.
- Downloads - Software development kits (SDK) provide resources to help users build, modify, test, and execute specific tasks.