# ALLJOYN® THIN CORE

ALLJOYN®精简内核

## 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.

概览

AllJoyn 是一套开源软件系统，它为可以夸不同类型设备运行的分布式应用提供运行平台，并且强调移动、安全和动态可配置的特性。AllJoyn是“平台不依赖”的，这意味着它在设计之时就尽量做到不依赖与特定的操作系统、硬件或硬件上运行的软件。

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.

AllJoyn的标准内核库(AJSCL)中的各子模块设计是考虑了在Microsoft Windows、Linux、Android、iOS、OS X、OpenWRT下以及作为互联网浏览器的整合插件的方式使用的要求。这些软件系统的一个共有的特性是它们都运行于通用计算机之上。通用计算机通常拥有可观数量的存储能力，有充足的电力、计算能力以及由此带来的能支持多进程、多线程和多语言环境的操作系统。

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.

与此不同，嵌入式系统运行于嵌入在一个大型设备上的微处理器当中以提供特定的功能。由于嵌入式系统只需要执行特定的任务或一些有限数量的功能，工程师可以优化这些系统以便能在具有更小的存储空间、更有限的处理器速度、更省电、更少的外设和用户接口等的硬件平台上运行来减小产品的尺寸与价格。AllJoyn Thin Core Library（AJTCL）就是以将AllJoyn分布式编程环境的优势应用于嵌入式系统为目标应运而生的。

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 the [**Introduction to the AllJoyn Framework**](https://allseenalliance.org/developers/learn/core/standard-core)) 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.

由于AJTCL将要运行的操作环境将会是非常受限的，运行于此类系统上的AllJoyn组建必须适应各类限制。具体来说，这就意味着我们将不具有AllJoyn router(需要多线程)所具有的奢侈条件，比如丰富的网络连接、使用相对大的RAM和ROM空间等。同时我们也不具备运行能够包含课更换语言关联的面向对象的编程环境的条件。在这样的条件下，AJTCL只考虑单独使用C语言来实现总线接入的功能（参见**[Introduction to the AllJoyn Framework](https://allseenalliance.org/developers/learn/core/standard-core)**）。与接口、方法、信号、属性以及总线对象所对应的数据结构都进行了高度优化以节省空间，开发者使用的API也因此有很大的不同。

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.

虽然API会有所不同，但是所有的AJSCL下的主要的概念性模块都可以在AJTCL中找到；它们更类似于以一种更紧凑的形式予以展现或者说实际上是在另一个远端能力更强的机器上运行而已。

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

注意：当我们提到AllJoyn Standard Library (AJSCL)时，我们更明确的说是指在通用计算机上运行的这些模块的版本。

## Conceptual Model

As implied in the previous section, most high-level abstractions used in AJTCL are identical to those in the AJSCL system. The [**Introduction to the AllJoyn Framework**](https://allseenalliance.org/developers/learn/core/standard-core#conceptual-overview) 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.

## 概念性模型

正如以上章节提到的，AJTCL中使用最高层次的抽象模型与AJSCL系统是完全对应的。在**[Introduction to the AllJoyn Framework](https://allseenalliance.org/developers/learn/core/standard-core" \l "conceptual-overview)**中有专门的章节Conceptual Overview(概念综述)为大家描述了这一抽象模型。在概念综述的章节中，我们假设读者已经熟悉了相关文档中对抽象模型的介绍，因此我们只会涉及理解AJTCL结构所需要的特使的部分。

### 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.

### AllJoyn Thin Core Library 仍然是AllJoyn

需要特别强调与注意的是AJTCL是AllJoyn框架的一部分。Thin Core Library具有完全的对AJSCL的互操作性。由于AllJoyn网络连接协议在两种系统下都有完整的实现，AJSCL是完全不用知道与之沟通的是否是一个Thin Core Library的，对于AJTCL也是同样。

Recall from the [**Introduction to the AllJoyn Framework**](https://allseenalliance.org/developers/learn/core/standard-core) that the basic structure of an AllJoyn distributed bus consists of multiple bus segments residing on physically separate host computers.

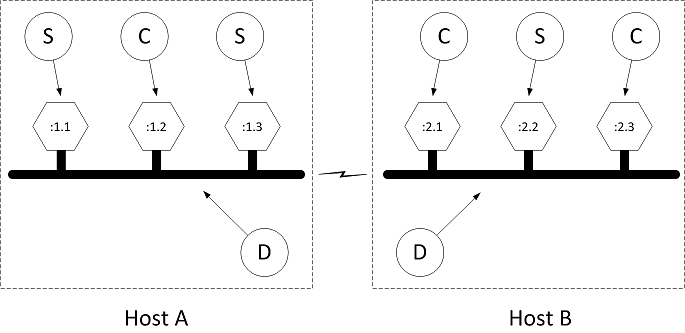


Figure: AllJoyn distributed bus

按照**[Introduction to the AllJoyn Framework](https://allseenalliance.org/developers/learn/core/standard-core)**的介绍，AllJoyn分布式总线的基本结构可以由挂靠于多个独立的物理主机上的总线段共同组成。

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).

每一个挂靠于特定主机的总线段在图中用标为Host A和Host B的虚线所围方框表示。每一个总线段都由一个AllJoyn路由器（图中标为D的圆圈所示）来实现。每个主机都可能有多个设备接入，连接至本地域（图中六边形所示），将它们分为服务端（S）和客户端（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.

由于运行AJTCL的设备通常没有资源运行路由程序，在AllJoyn的结构中连接到分布式总线上的Thin Core Library 借用主机上运行的路由程序。

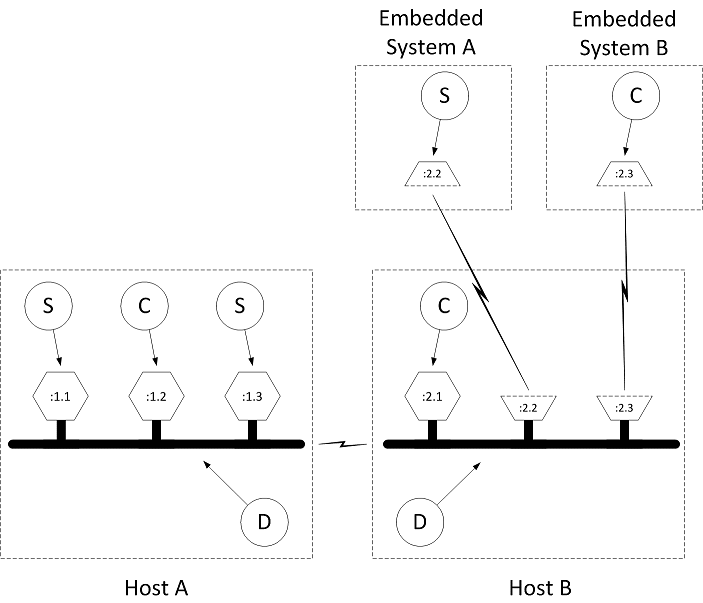


Figure: AllJoyn distributed bus with thin core libraries

图： AllJoyn 下连接Thin Core Libraries的分布式总线

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).

需要注意的是嵌入式系统A和嵌入式系统B与他们所属的运行用来管理分布式总线段的路由程序的主机B是不同的设备。

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**](https://allseenalliance.org/developers/learn/core/standard-core).

如**[Introduction to the AllJoyn Framework](https://allseenalliance.org/developers/learn/core/standard-core)**所述，嵌入式系统与路由设备之间通过网络传输的是AllJoyn的各类消息，用来实现总线方法、总线信号以及属性等在各自会话内的交换。

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).

在有些应用中，会有要求AJTCL设备连接或借用已存在的近场路由的需求，我们称之为非信任关系（从路由角度来说）。同事也会有允许特定的AJTCL设备连接特定的路由的需求，我们称之为可信任关系（同样从路由角度来说）。

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").

这些关系是通过使用发现与连接流程来实现的，概念上讲这与服务端与客户端的发现与连接流程是类似的。一个AllJoyn路由设备会主动向一组AJTCL设备发布其公开的命名表示其接入的能力。这一广告由路由配置或是一个特定的AllJoyn广告组件来驱动。当向一个路由设备发起一次连接请求时，会产生一个发现事件，对于期望信任关系的路由服务，可能会选择向特定的Thin Core Library（或者Thin Core Library的模仿者）提出询问以产生一次鉴权。对于非信任关系的情况，路由设备有可能直接选择允许任何连接请求，而在此情况下，相关的路由服务不会允许Thin Core Library进行任何会引起与非本地设备建立会话的操作（即“产生费用的服务”）。

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**](https://allseenalliance.org/developers/learn/core/standard-core), 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.

如上所述，对于AJTL设备而言连接的过程可以被分为三个阶段：

* 发现阶段
* 连接阶段
* 认证阶段

发现阶段工作原理与在 [**Introduction to the AllJoyn Framework**](https://allseenalliance.org/developers/learn/core/standard-core)中描述的服务广告和发现一样，但是两个区别。第一个区别是以AJTL发现为目的的广告是“安静”模式的，简单讲也就是说这类广告不是有路由器无偿发放的。

第二个区别是对于安静广告的回应是安静的，我们称之为安静回应。也就是说回应将会被单独直接发回给请求者而不是类似于“主动”广告那样广播出去。这样的改变的主要原因是允许没有实现多播回应的嵌入式设备参与到AllJoyn分布式系统中来。

### 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:

### 什么是AllJoyn Thin Core Library设备？

一个典型的AJTCL设备概念上来讲类似于一个无线传感器网络中的传感器节点。传感器节点通常是能量、计算能力、存储能力或其它资源均受限的小尺寸传感器或伺服器。它们能够感知周围的情况，与外界沟通发生的事件，或者有可能根据内部的处理机制或外部事件的结果采取特定的行动。这非常依赖于设备的功能定义，能归为此类定义的设备我们可以举一些其中很小一部分的例子：

* Light switches
* Thermostats
* Air conditioners
* Vent dampers
* Smoke detectors
* Motion detectors
* Humidity detectors
* Microphones
* Speakers
* Earphones
* Doors
* Doorbells
* Ovens
* Refrigerators
* Toasters

电灯开关

恒温器

空调

排风扇

烟雾传感器

运动传感器

湿度传感器

麦克风

扩音器

耳机

门

门铃

微波炉

冰箱

烤箱

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**](https://allseenalliance.org/developers/learn/core/standard-core).

关于传感器网络有着大量的文献进行讨论。与WSNs中不太看重安全性的典型使用的自组织、多跳自组网的无线网络不同，AllJoyn架构通常运行于参与设备都是相关的且经过认证的Wi-Fi基础模式的网络之上。为了完成对Wi-Fi网络的安全认证，AJTCL使用了称为“onboarding（登录）”的流程。登录服务架构使通常不具有友好用户界面的Thin Core Library设备能够获得足够的目标网络的信息来完成加入该网络所需的许可与鉴权的过程。登录服务的架构在专门的章节中有详尽的描述。

作为传感器节点在AllJoyn的场景中AJTCL设备通常实现一项服务。它使用所依赖的硬件感知周围的情况并且将事件以AllJoyn信号的方式发送给外界，并根据由其他设备发送的事件或响应AllJoyn客户端的远程方法调用触发一次动作，如**[Introduction to the AllJoyn Framework](https://allseenalliance.org/developers/learn/core/standard-core)**中所描述的那样。

## 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 AJSCL or service process is reproduced below.

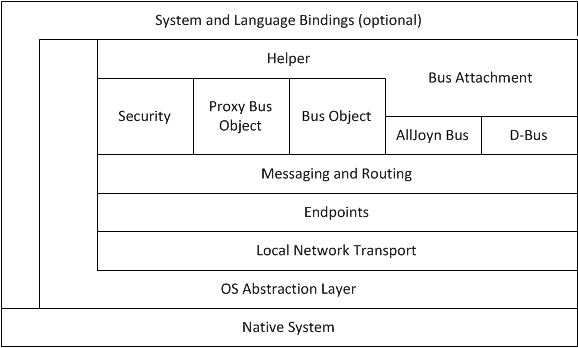


Figure: AJSCL layering

## Thin Core Library 结构

由于AllJoyn Thin Core Library (AJTCL)必须运行于能源、处理能力及内存均受限的设备上，没有足够的资源使用与通用计算机系统上运行的AllJoyn 标准内核库 (AJSCL)类似的结构。

层次化的AJSCL或服务流程如下图所示。

See the [**Introduction to the AllJoyn Framework**](https://allseenalliance.org/developers/learn/core/standard-core) 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.

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 below.

更详细的各层的讨论可以参考**[Introduction to the AllJoyn Framework](https://allseenalliance.org/developers/learn/core/standard-core)** 。

在此需要注意的非常重要的一点是每一个AllJoyn分层中的客户端或服务端都对应于一个AllJoyn流程上的应用。

每一个具有AJSCL主机都具有至少一个AllJoyn路由服务。这一服务在独立路由的情况下只有自己一个进程，在集成路由的应用情况下也可能和应用程序联合部署。分层次的AJSCL路由结构如下图所示。

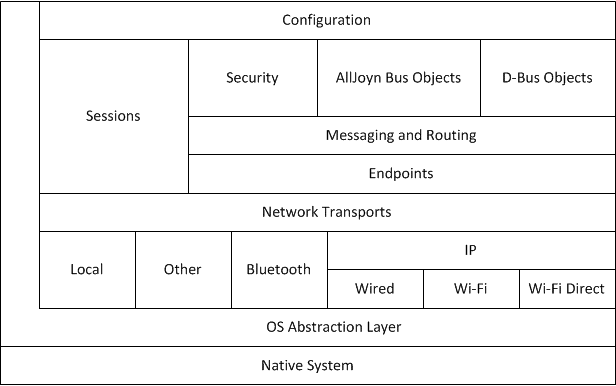


Figure: AJSCL router layering AJSCL路由层次结构

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.

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 below, does away with much of the overhead present in the AJSL system.

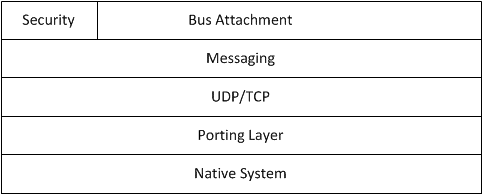


Figure: AJTCL layering AJTCL层次结构

需要注意的是路由功能还提供在路由器之间转发消息的额外支持，并具有使用多网络传输机制如Wi-Fi直连的能力。实现这些功能所需的计算能力、能耗和存储代价将是非常可观的。

显然如此大量的代码是无法在受限的嵌入式系统上运行的，因此AJTCL精简了大量的代码只保留了必须存在的部分以使其能运行于所需的设备之上。为此基本的运行环境仅限于最小化的实时C语言环境，并借用其他设备来完成路由的角色。与AJSCL不同，AJTCL去除了AJSL系统中很多上层表述性部分，如下图所示。

AJTCL exposes only the minimum required API to the bus attachment and exposes the AllJoyn messaging interface directly instead of providing helper functions.

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.

AJTCL只向总线接入程序开放了尽可能少的所需的API，并直接开放了消息接口而非帮助函数。

消息层没有使用抽象传输机制而是直接使用了UDP和TCP。一个非常简易的接口层用来提取个别所需的原生系统函数，整个包都是使用C语言编写的以保证最小的代码尺寸。由于优化的结果，AJTCL系统可以在仅有20Kbybtes存储空间的系统上运行，而一个路由服务加上C++的客户或服务端的集合将会需要10倍的存储空间，Java语言的版本所需空间甚至达到了40倍。

## Tying it All Together

In order to make this discussion somewhat more concrete, two example distributed systems are presented here.

* 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.
* 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.

**系统集成**

为了使这里的讨论更直观具体，我们这里举两个分布式系统的例子：

* 一个运行单个AllJoyn应用的智能手机上的最小系统与一个AJTCL设备互通。这个例子用来说明上文中描述的课信任关系。
* 一个无线路由器上运行路由服务的相对复杂的系统。

注意：通常来讲实际情况是一个运行OpenWRT的路由器上有一个预装的AllJoyn路由服务。这一路由接受来自于已经登录到Wi-Fi网路的Thin Core Library的非信任连接。

一些AJTCL设备连接到路由器作为基于AllJoyn的无线传感器网络的传感器节点，由一台通用计算机执行数据融合功能。

注意：在无线传感器网络中，数据融合专指一些不同的节点从一定数量的传感器节点上收集结果并进行集成或“融合”，然后依据这些传感器节点上得到的数据产生相应的结果或需要执行的动作。

### 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.

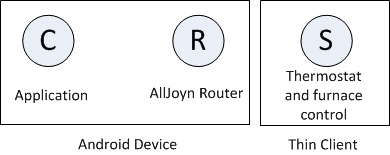


Figure: Minimalist example system 最小系统例子

### 一个最小化的Thin Core Library系统

一个使用AJTCL的最小化系统的例子包括一个运行AJSCL的单主机和一个Thin Core Library设备。AJSCL 提供了Thin Core Library连接的AllJoyn路由器，并且提供一个运行使用Thin Core Library的应用的平台。如上面所述，Thin Core Library通常作为一种传感器节点使用，向主机上运行的应用发送数据。应用程序以某种方式处理这些数据并向传感器发送命令以更改其环境参数。

一个简单而合理的系统，例如墙上有一个控制锅炉的恒温器，Andriod设备上有一个用于控制的应用程序。AJSCL会运行在Android设备上而墙上的恒温器内运行AJTCL。

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

在这个例子里，一个需求是墙上的恒温器只能被Android设备上对应的恒温控制程序控制。

由于例子中恒温器只能被Android设备上对应的恒温控制程序控制，所以恒温器很可能要求与Andriod上的应用所使用的路由器建立连接。这意味着Android应用应该与AllJoyn路由器绑定并且只有这一特定的绑定的路由器/应用对才能以路由器的能力告知Thin Core Library来使用。这类应用导致AJTCL和路由器/应用对之间的可信任关系。

之后应用会请求绑定的路由器以安静模式公示一个AJTCL可见的公开名（例如com.company.BusNode

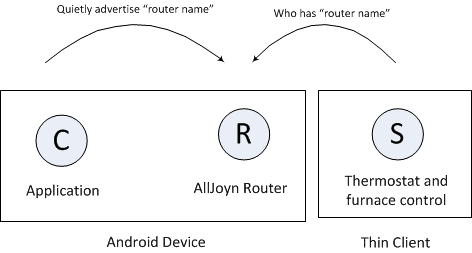


Figure: Thin Core Library router discovery Thin Core Library路由器发现Thin Core Library

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.

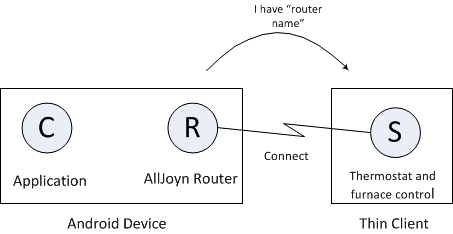


Figure: Thin Core Library connection attempt

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**](https://allseenalliance.org/developers/learn/core/standard-core), the arrangement appears as if the AllJoyn router has a connected service and client.

当路由器收到了一个对它以安静模式广播的公开名的准确的询问时，它会回应一个标示表明需要的公开名确实连接到了这一特定的路由器上。之后AJTCL会试图连接到有回应的路由器上。

至此一个逻辑上的AllJoyn总线已经形成了，应用程序与Thin Core Library服务都连接到了绑定的Android设备上的路由器上。使用[**Introduction to the AllJoyn Framework**](https://allseenalliance.org/developers/learn/core/standard-core)中的圆圈图来表述这一系统如下图所示，这样的连接与路由器已与服务端和客户端建立了连接的效果一致。

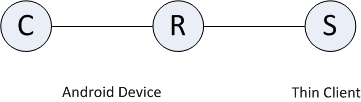


Figure: Thin Core Library system example

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**](https://allseenalliance.org/developers/learn/core/standard-core) 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.

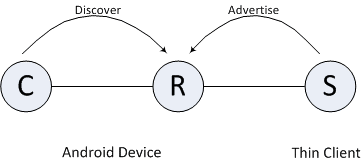


Figure: Service discovery with the Thin Core Library

Thin Core Library上的服务发现

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.

此时AJTCL已经和与应用绑定了的路由器建立了连接，但是应用和AJTCL都不知道彼此的存在。通常此时AllJoyn场景中AJTCL会请求一个公开的总线名并且实例化一个服务。如[**Introduction to the AllJoyn Framework**](https://allseenalliance.org/developers/learn/core/standard-core)所描述的Thin Core Library会创建一个会话接口并且使用Thin Core Library APIs广播一个公开名。这一公开名通常与绑定的路由器广告的公开名不同；它对应于Thin CoreLibrary与应用程序间的客户端/服务端的关系而不是路由器与Thin Core Library间的关系。然后Android设备上的应用会进行这一公开名的服务发现。

当AJTCL上的服务被Android设备上的客户端发现后客户端可以加入由服务端建立的会话。

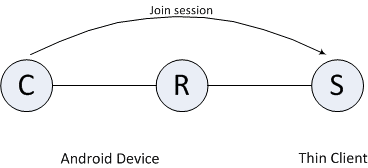


Figure: Android device joins session with service on the Thin Core Library

Android设备加入Thin Core Library上服务的会话

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**](https://allseenalliance.org/developers/learn/core/standard-core).

至此Android设备上的应用可以像所有AllJoyn设备一样的访问AJTCL上的服务了。它可以选择接收服务端发起的通知信号，这个例中的情况，可能是定期收到当前的温度信号。应用也可以显示一个允许用户输入想要温度的用户界面，然后将新的目标温度以AllJoyn架构描述中介绍的远程方法调用（RMI）的方式发送给AJTCL端。当收到一个方法调用时，AJTCL上运行的服务将会把请求转发给锅炉来设定目标温度。

Thin Core Library一侧使用的API与AJSCL或服务端使用的有很大的不同，但是连接协议是一致的，所以连接的另一端（AJSCL或AJTCL）组件的类型是不可见的。从这个角度来说，在 [**Introduction to the AllJoyn Framework**](https://allseenalliance.org/developers/learn/core/standard-core)中圆圈图所表示的所有AllJoyn节点包括AJTCLs对于所有的使用意图与目的都是不可区分的。

### 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 below.

### 基于Thin Core Library的无线传感器网络

这个例子是一个非常基本的家庭管理系统。假设无线接入点是一个预装了AllJoyn路由服务的能支持Thin Core Library非信任连接的OpenWRT路由器。这使得所有接入了这一系统的ATJCL可以连接到路由服务域内。这一网络内的Thin Core Library可能是温度传感器、运动探测器、灯光开关、热水恒温器、锅炉或空调系统的温控器。

如上所述，例子网络中的数据融合功能是有一个通用计算机上的具有集成显示功能的应用程序执行的。网络中并不是要求一定要有专门的通用计算机，数据融合任务也可以分布式地去完成，但是网络中有这样的组件使我们能够更好地描述与讲解AJSCL和Thin Core Library设备之间是如何互动的。“融合器”的显示器可以安装在家里的墙上，或者就是家里某处的一台个人电脑。这个显示器可以提供一个用户界面以显示例如各个房间的温度的信息，或者虚拟灯光的开关，或者运动探测器。实际的数据融合功能算法应该判断何时去开灯关灯，打开暖气、锅炉，或者在什么时候调整热水器的温度以达到最佳的效率等等。

首先需要考虑的模块是OpenWRT上的路由器，如下图所示。

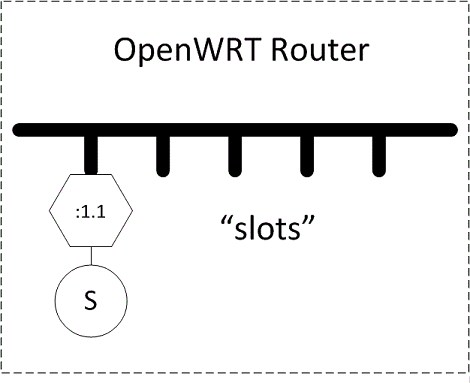


Figure: OpenWRT router hosting a standalone AllJoyn router daemon

OpenWRT 路由器运行独立的AllJoyn路由域

The router hosts a standalone AllJoyn router daemon, and is illustrated as the bold horizontal line that represents a segment of an AllJoyn distributed software bus.

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.

路由器上运行AllJoyn独立的路由域在图中以加黑的横线表示，代表AllJoyn分布式软件总线的一个段。

路由器总线段所挂靠的AllJoyn服务中有可能有服务能够提供一种方式对路由器和使用AllJoyn架构的路由服务本身进行配置。除此之外还有一些空的插口代表连接到AJTCL的非信任连接。由于这是一个一般性的AllJoyn路由器，对应的软件总线可以扩展到其它的总线段以形成一个分布式的总线。

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.

AJTCLs 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.

在之前的部分提到过，AJTCL设备会主动以发现的方式搜索可以连接的路由设备。这里我们描述的是非信任关系，因此运行于OpenWRT上的AllJoyn路由器会根据配置安静广播通用名如org.alljoyn.BusNode来表示这个路由器是一个AllJoyn分布式总线上具有Thin Libraries接入能力的节点。

分布式网络中代表着传感器节点的AJTCL会通过登录过程加入到这一无线网络中来。在这个过程中，它们会被赋予一个称为友好名的标识以表明其在家庭中的使用意义。比如一个灯光开关伺服器（开关亮度控制）可以被命名为“厨房”或是“起居室”。相对应的Thin Core Library节点会开始发现它们被分配的路由器（比如org.alljoyn.BusNode）然后试图建立连接。由于OpenWRT上运行的预装路由器处于非信任模式下，来自Thin Core Library的连接会被接受。

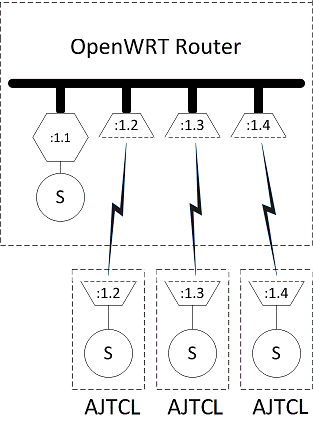


Figure: AJTCL nodes connected to the OpenWRT AllJoyn router

ATJCL节点连接至OpenWRT上的AllJoyn路由器

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.

一旦Thin Core Library应用于OpenWRT路由器上的总线段建立起了连接就会开始发布其所具有的服务特性。这里假设已经有一个家庭控制系统登录到了这一路由器提供的无线网络中，它就会开始发现并寻找系统中由Thin Core Libraries提供的服务。

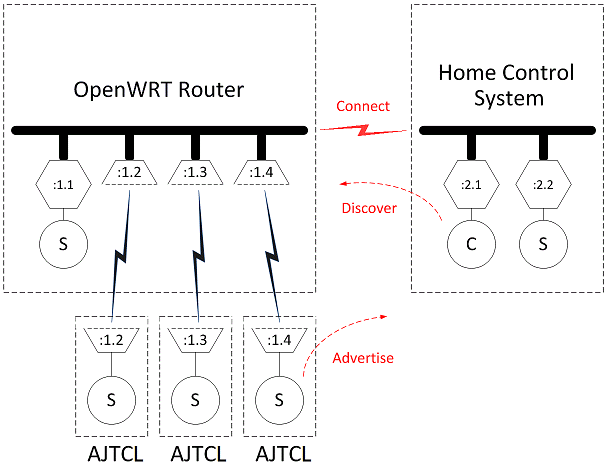


Figure: OpenWRT router, Thin Core Libraries, and home control system

OpenWRT路由器，Thin Core Libraries和家庭控制系统

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**](https://allseenalliance.org/developers/learn/core/standard-core). This will result in the bus segments implemented on the router and the home control system merging into a single virtual distributed bus.

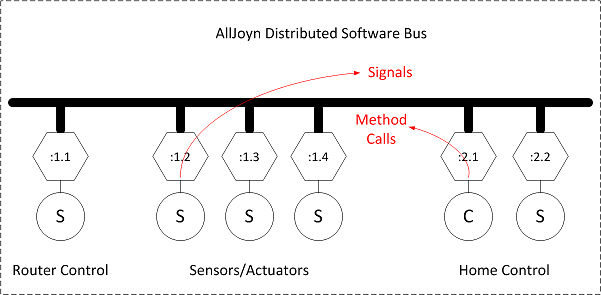


Figure: AllJoyn distributed software bus

AllJoyn分布式软件总线

和在[**Introduction to the AllJoyn Framework**](https://allseenalliance.org/developers/learn/core/standard-core)中讨论的一样，一旦家庭控制系统发现了AJTCL设备中某一个所发布的服务，他就会试图加入与发现的Thin Core Library的会话中去。这样就使得路由器和家庭控制系统上分别实现的总线段融合成为了一个完整的虚拟分布式总线。

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.

当融合后的总线完全形成以后，挂载在这一总线上的设备都会提供公开的AllJoyn客户端或服务端。通过TCP连接到AllJoyn路由器的 Thin Core Library的传感器和伺服器实际上是一些嵌入式设备，并不直接暴露给分布式总线上的其它组件。而家庭控制系统有可能是运行在通用计算机上的Andriod系统上的Java程序，也不会直接暴露给分布式总线上的其它组件。客户端与服务端都简单地实现远程方法调用并交换信号。

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.

至此数据融合器上运行的算法已经可以被很清楚的理解了。一个通过分布式总线发送的重要的AllJoyn信号，比如CARBON-MONOXIDE-DETECTED（发现一氧化碳）应该能够被家庭控制系统（数据融合器）接收到，之后后者作为响应会发送一个远程方法调用给一个伺服器节点以打开风扇，同时会发送另一个调用给另一个节点来触发警报，还可能向房主发送一条短信告诉他房间里发现一氧化碳超标。

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.

更平常的家庭控制系统的功能可能是当家里没有人在的时候（由运动探测器报告或按照提前设定好的日程规划）向锅炉发送远程调用来降低屋内的温度。工作日中或深夜时，家庭控制单元可以向热水器发送一个消息来降低水温，同时加热洗碗机的用水使它在电费最低的时候运行。

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.

所有的这些家庭控制系统的反应和方法调用信号都完全与消息的来源和目的地设备的类型和位置没有关系。

## 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.

总结

AllJoyn是一个在异质系统上部署分布式应用为目标的可以广泛使用的系统架构。AJTCL使嵌入式系统可以参与到AllJoyn的分布式总线中来并且向系统中的其它组件以抽象服务的方式公开，而这对于以往的开发者来讲在这类异质性的系统上去实现将是非常困难的。这一途径使应用开发者可以专注于具体的内容应用的开发而不需要大量的底层嵌入式系统或网络的开发经验。

AllJoyn系统是作为一个整体进行设计的，不会有ad-hoc系统中那样从大量碎片拼凑在一起时遇到的继承性的痛苦。我们相信AllJoyn系统可以使有嵌入式系统模块参与的分布式应用的开发与部署相较于其它平台更为简单高效。

## Learn More

To learn more about how to integrate the AllJoyn framework in your development efforts, access the documentation and downloads available on the **[AllSeen Alliance web site](https://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.

## 获取更多信息

如想获取更多关于如何将AllJoyn架构集成到你的开发项目中的信息，请访问[**AllSeen Alliance web site**](https://allseenalliance.org/)以获得相关文档或下载内容。

* Introductory guides（介绍导读） - 描述AllJoyn技术与概念
* Development guides（开发导读） - 提供建立编译环境的步骤并提供特定编程问题的解决方案包括源代码片段与详细解释。
* API references （API 参考）- 提供使用AllJoyn源代码的详细说明以及使用各个支持的语言编写的应用。
* Downloads （下载）- 软件开发套件 (SDK)用以提供用户编译、更改、测试以及执行特定任务所需的资源。