Expand Device Capability with Multiprotocol Bluetooth and Zigbee Connectivity

Multiprotocol connectivity provides a unique approach to add functionality being requested by consumers and businesses. To provide the necessary scalability and robustness in home or building automation scenarios device-to-device communication over a mesh network is an ideal implementation. At the same time being able to setup, control, or monitor an individual device or a set of devices directly from a smartphone is a feature being requested to simplify consumer experience and provide more immediate diagnostic information to technicians for installation.

Table of Contents:

Value added services such as providing proximity based advertisements in retail settings, transmitting system health information for technicians, and tracking assets in warehouses can be delivered through connected devices such as lights. At the same time, there is a desire to participate in multiple ecosystems whether it's Alexa, Apple HomeKit or Google Home, each with their own protocols or integration requirements. By supporting multiple protocols on an individual device we are able to address a number of the needs we just discussed.

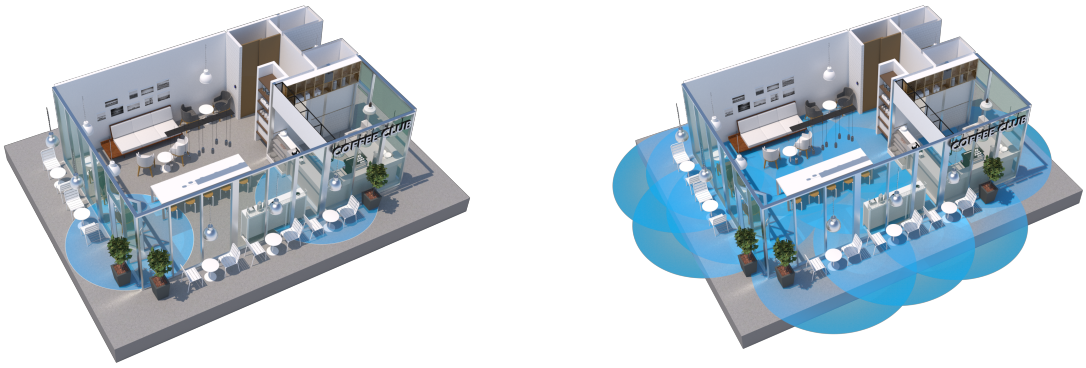
Providing New Experiences with Multiple Wireless Protocols

Let's examine how the experience in home automation scenarios can be improved with devices that support multiple protocols. Zigbee provides whole home coverage with its mesh capabilities and can provide control from outside the home via a gateway. However, with multiprotocol support, we can further expand the use case to include the phone, which has Bluetooth low energy, for local control and location aware services.

By supporting both Bluetooth and Zigbee connectivity the door lock unlocks after receiving Bluetooth communication and is simultaneously able to send a Zigbee message to turn on the living room lights. Using proximity aware services such as Bluetooth beacons when the smartphone is brought into the bedroom, the lights can send a Bluetooth beacon message to allow the consumer to turn on all or some of the lights in the room.

In retail or commercial settings, there is a desire to make use of technologies such as Bluetooth Beacons to provide location based advertisements, track assets and also develop heat maps of foot traffic. One of the challenges for wide scale adoption is the need for dedicated beacon devices. For device life cycle management, the range of connectivity also has impact on the logistics of updating devices.

By integrating Bluetooth beacons into other connected infrastructure such as lighting we can build out large scale and dense beacon coverage areas. Instead of having to deploy both connecting lights and beacons, a connected Light or Luminaire can also serve as the Bluetooth beacon. This can provide a more cost-effective means to improve beacon density than deploying separate dedicated Beacon devices with an added advantage of not requiring battery powered beacon devices that must be monitored and maintained.



Improving beacon density with multifunction lights

Multiprotocol makes additional uses cases possible as well. For example, over-the-air updates can take a long time over a mesh network, but the higher throughput of Bluetooth can provide quicker transfer of update images without consuming the bandwidth of the mesh network.

Finding a Cost-Effective Means to Support Multiple Protocols

One of the challenges faced to deliver these improved experiences with support of multiple protocols has been the requirement to have multiple chips or SoCs, one for each protocol. With multiprotocol chips, devices now have the flexibility to run different protocols. The below table describes some common examples of multiprotocol devices

Mulltiprotocol Type	Description
Programmable	Devices are programmed to run one of a number of protocols in manufacturing or via updates in the field or over the air.
Switched	Devices change which wireless protocol is being used by bootloading a new firmware image when the device is already deployed in the field.
Dynamic	Devices simultaneously run multiple wireless protocols on one chip, using a time-slicing mechanism to share the radio.
Multiradio	Devices achieve dedicated operation of multiple protocols without any trade-offs with multiple radios.

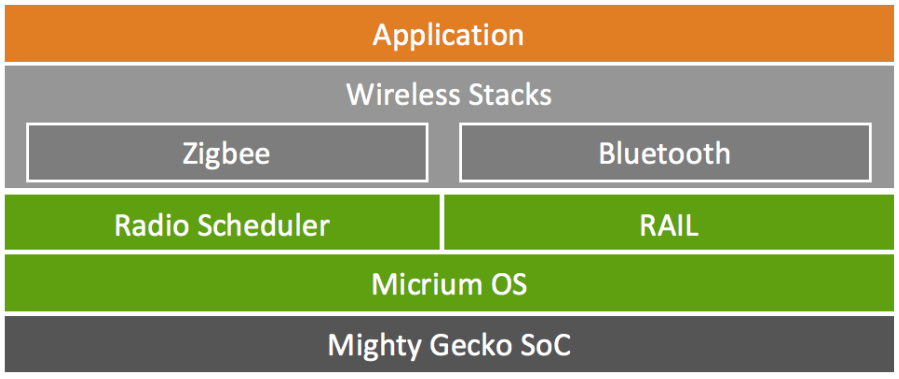
Table 1: Wireless Multiprotocol Scenarios

Single chip solutions that combine advances in software and hardware from companies like Silicon Labs enables devices to support both Zigbee and Bluetooth to address the use cases discussed so far. By using one SoC wireless sub-system BoM cost can be reduced by 40% by eliminating two radios and also simplify PCB design by eliminating the need to address the possible interference between two radios in a design.

Simultaneous Execution of Multiple Protocols on a Single Radio

Let's take a more detailed looked at how dynamic multiprotocol scheduling works to support multiple protocols with a single radio. A ZigBee router always has its radio in receive mode when not transmitting. This is so that other devices in the network can always send packets to it, or route through it. Because of the low duty cycle for ZigBee traffic and the retry mechanisms in the ZigBee networking stack, it is possible for the ZigBee Router to change its radio to another protocol for short periods without dropping any messages at an application level. This allows us to time-slice Zigbee and Bluetooth communication on the same chip. In addition to Zigbee routing, Silicon Labs dynamic multiprotocol technology supports Bluetooth connections, and Bluetooth beacons.

The protocol connection interval is configurable to match application requirements. For Bluetooth beacons, the radio only needs about 1ms to transmit a beacon and the connection interval between beacons is typically no shorter than 100ms. For high speed OTA firmware updates, the device will likely need to be configured to support much longer Bluetooth connection periods. These examples lie on opposite ends of the spectrum; however, with a configurable connection interval, the multiprotocol solution from Silicon Labs provides a flexible framework to meet the unique needs of different applications.



To enable effective multiprotocol communication Silicon Labs has made a number of investments in both software and hardware. Wireless protocol stacks from Silicon Labs are architected to share the same low-level radio drivers and libraries (RAIL). Making use of RAIL ensures a consistent API and interface to share the radio.

In addition, the Radio Scheduler manages the requests from protocols for access to the radio while the Micirum OS kernel manages the sharing of resources between the stacks.

The multiprotocol scheduling from Silicon Labs takes the protocols being scheduled into account and uses a priority based scheduling methodology. Bluetooth requires a fixed connection interval for effective operation while Zigbee, with its MAC retry approach is more forgiving. For this reason, for Zigbee and Bluetooth multiprotocol operation, Bluetooth operates at a higher priority. Because of the unified architecture of wireless stacks using RAIL, radio scheduler and Micirum OS the system is able to use a priority based scheduling approach to balance Zigbee and Bluetooth operation.

Scheduling Requirements for Single Radio Zigbee and Bluetooth Operation

A number of scheduling scenarios may be required to enable the correct operation of Zigbee and Bluetooth with a single radio. The scheduler can be configured to make either protocol the higher priority with regards to radio access. However, the most likely configuration would be to make Bluetooth connections and beacons the higher priority and leave the radio in Zigbee receive mode when doing nothing else.

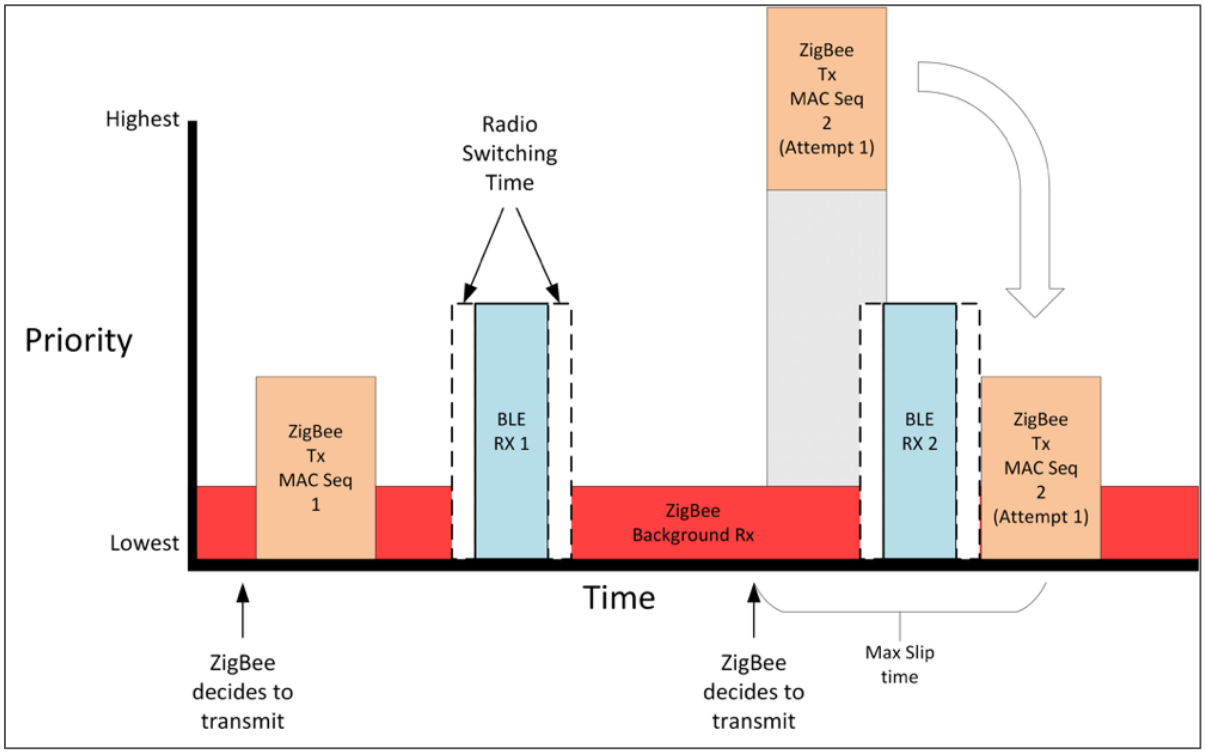


Figure 1: Zigbee background receive with Bluetooth LE having priority

In Figure 1, we can see that the low-priority Zigbee receive is the default, but when a Zigbee transmission is required, it interrupts that process. This is normal behavior for a Zigbee device. When a Bluetooth LE connection is scheduled, this takes precedent, and the scheduler switches out of Zigbee receive mode in time to be available for the Bluetooth connection. If the scheduler has a request for a Zigbee transmission that would exceed the time available on the radio before the next Bluetooth connection or beacon, the scheduler will reschedule the Zigbee transmission to occur after the Bluetooth activity has completed.

If the transmission time for a Zigbee packet exceeds what was expected, perhaps due to backoffs or clear channel assessment, the scheduler can interrupt that transmission and switch to Bluetooth. To the Zigbee stack, this looks like a failed attempt, so it retries the transmission, and this time it succeeds, as shown in Figure 2.

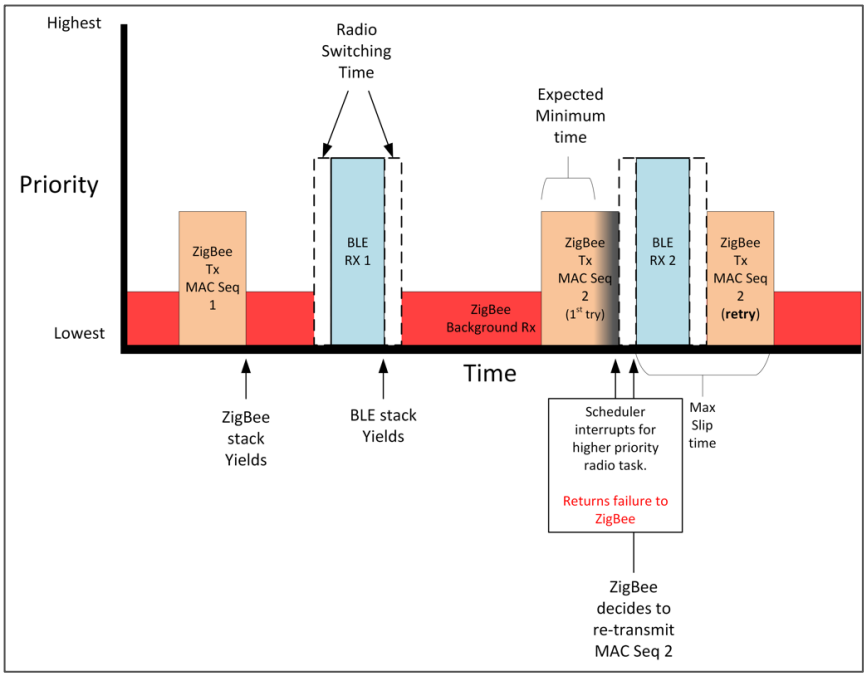
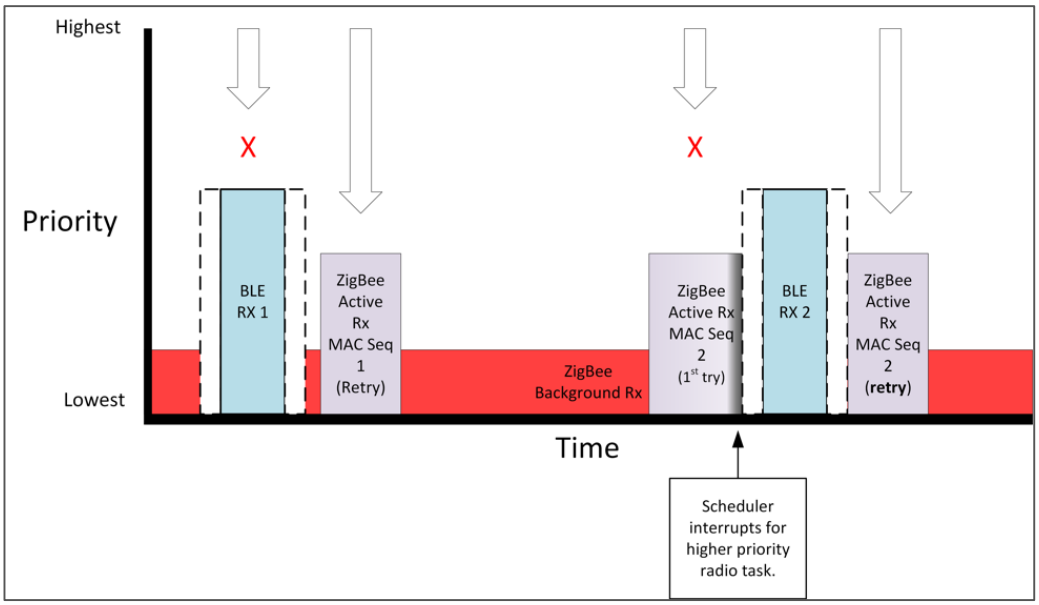


Figure 2: Bluetooth connection interrupts Zigbee transmission

Similarly, if a remote Zigbee node attempts to send a packet to the device while it is in the middle of a Bluetooth connection or beacon, the device cannot receive the packet, but the sending device will retry (IEEE 802.15.4 MAC retry), and the packet will be received on the second attempt. Also, if the device is in the middle of receiving a Zigbee packet when a Bluetooth connection or beacon is due, the scheduler can interrupt the packet reception, and the sending device will not receive an acknowledgement. As a result, it will retry the transmission and succeed on the second attempt. Figure 3. shows both scenarios.



The radio scheduler must handle a variety of scenarios to manage conflicts between wireless protocols, but the individual protocol stacks do not have any awareness of one another, only that they must request access to the radio and whether their transmission or reception has been successful.

For additional radio scheduling examples please refer to the Dynamic Multiprotocol Users Guide.

Evaluating Dynamic Multiprotocol Performance

In order to understand device behavior when running multiple protocols, it is important to measure and compare performance under multiple configurations. For the case of Zigbee and Bluetooth running on the same SoC and single radio the scenarios may include:

- Zigbee throughput vs Bluetooth connection(s) and/or advertisement intervals
- Zigbee latency vs Bluetooth connection(s) and/or advertisement intervals
- Zigbee throughput or latency vs varying Bluetooth packet types and sizes
- Zigbee retries and network behavior for varying Bluetooth connections and/or advertisements

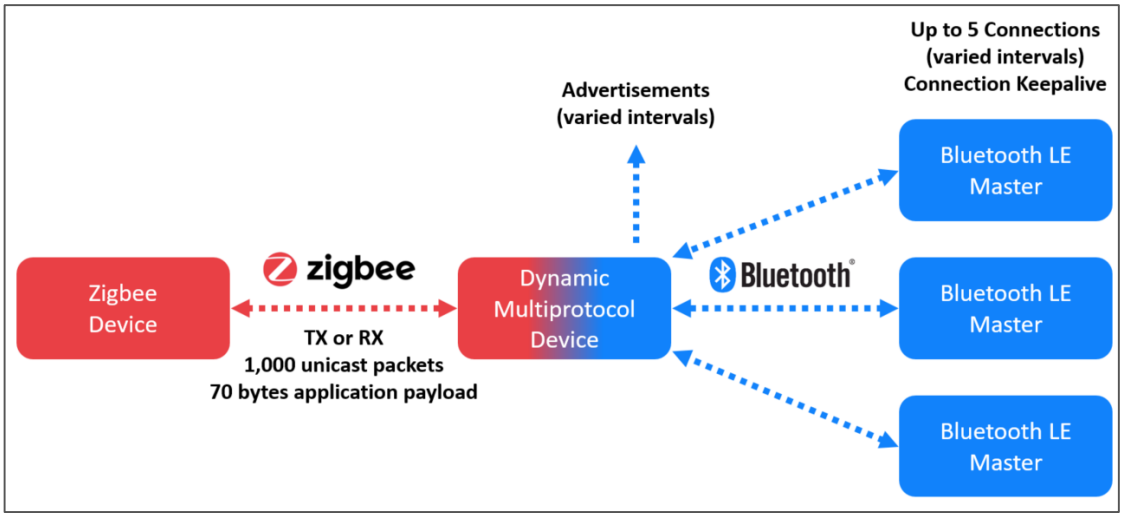
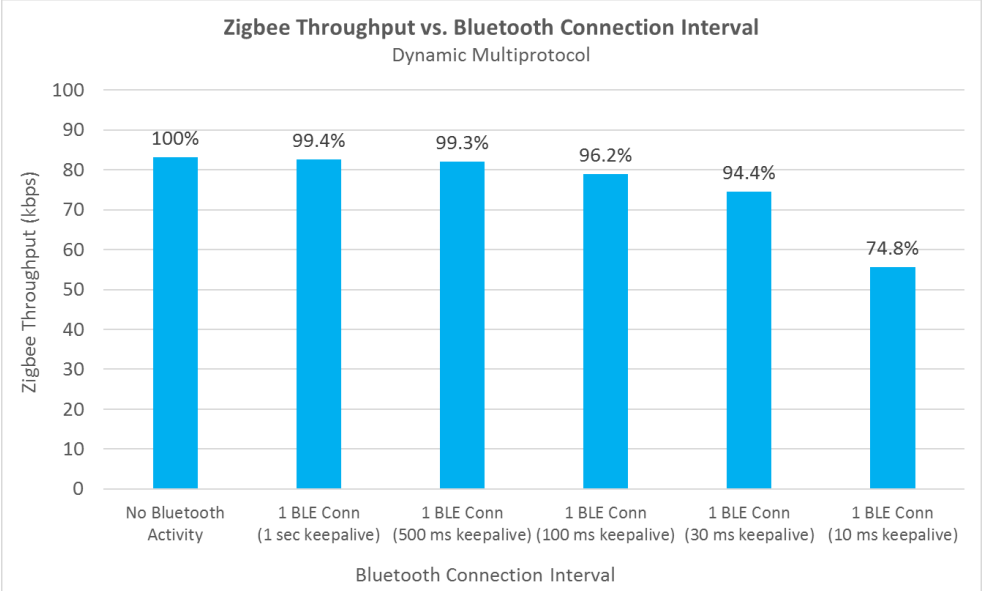


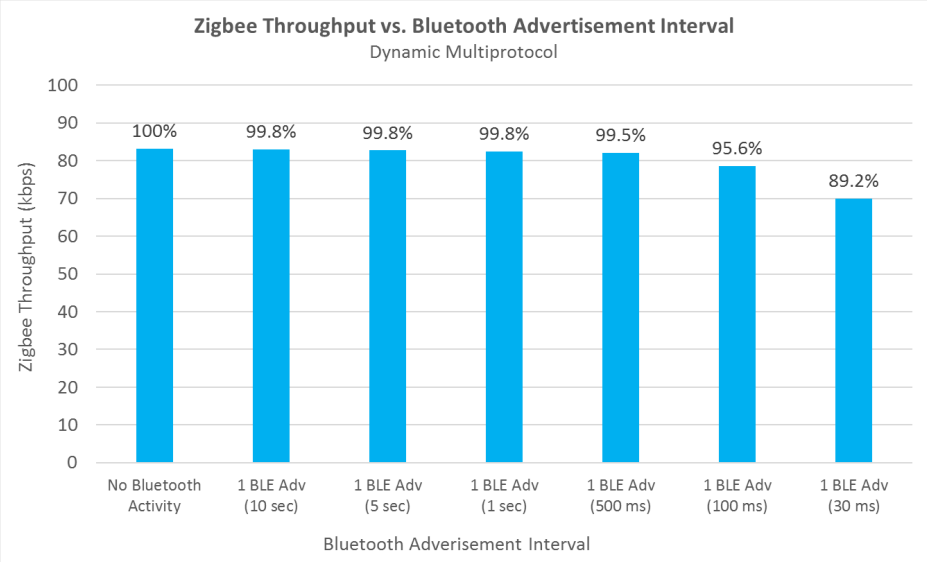
Figure 4: Dynamic Multiprotocol Test Setup

Using the test setup outlined in Figure 4 a sample test executed on a Silicon Labs Wireless Gecko STK board using a radiated test setup gives the following results:

For the results displayed 802.15.4 MAC and Zigbee NWK layer retries were enabled while Zigbee APS layer retries were not. The device was configured to transmit 70 bytes of payload across one hop while the Bluetooth connection was maintained with a keepalive at the noted connection interval. As the Bluetooth connection interval is reduced, the number of Bluetooth connection events increase and the Zigbee throughput decreases due to the decreased radio time on the Zigbee network. Note that 100% end-to-end message reliability was achieved and, although throughput decreased due to longer data transfer, no Zigbee application messages were lost.



To verify the impact of advertisement intervals the device was configured to transmit Bluetooth advertisements at varying intervals instead of maintaining a Bluetooth connection. Since Bluetooth advertisements packets are larger than Bluetooth connection keepalives, they have a slightly higher impact on Zigbee throughput for the same time interval. Advertisement intervals as short as 0.5 seconds have little impact on Zigbee throughput and should meet the needs of most use cases.



Designing Systems with Multiprotocol Connectivity

With dynamic multiprotocol hardware and software it is now feasible to combine the benefits of multiple protocols in cost effective manner on a single SoC. Home automation, asset tracking and retail advertising can benefit from combining Zigbee and Bluetooth connectivity on a device.

Each device and application has unique needs that require software configurability of items such as Bluetooth connection intervals. Before embarking on development is important to ensure the underlying software and hardware architecture is designed for effective resource sharing of the radio and enables advanced scheduling scenarios. In addition, testing and performance benchmarks should be defined with the specific application and system use cases in mind to ensure proper operation in the field.

To learn more about multiprotocol connectivity, click here.