Building a Sensor-Driven Lighting Control System Based on Bluetooth Mesh

***Bluetooth*® White Paper**

- **Revision:** v1.0
- **Revision Date:** 2020-08-25
- **Group Prepared By:** Mesh Working Group

Abstract:

Bluetooth mesh networking technology allows for building flexible, smart lighting networks that use sensors to drive efficiencies. One of its specifications—the Bluetooth Mesh Model Specification [1]—specifies several mesh models defining basic functionalities of network nodes. This white paper explains which mesh models to use in different types of lighting and sensor devices. It also describes how these devices can be set up to operate in concert and to form adaptive, highly efficient, connected lighting networks.

Revision History

Revision Number	Date	Comments
v1.0	2020-08-25	Approved by the Bluetooth SIG Board of Directors.

Contributors

Name	Company
Szymon Rzadkosz	Silvair, Inc.
Szymon Slupik	Silvair, Inc.
Piotr Winiarczyk	Silvair, Inc.
Jerzy Nosek	Silvair, Inc.
Laurence Richardson	Qualcomm Technology, Ltd.
Victor Zhodzishsky	Cypress
Khaled Elsayed	Synopsys
Marius Munder	Silicon Laboratories
Matthias Kassner	EnOcean

Contents

1 Introduction

This white paper illustrates how to set up an adaptive lighting system using the Bluetooth mesh networking technology. The paper first explains how to build individual devices that have mesh connectivity, starting with the most basic concepts and progressing to more sophisticated arrangements that introduce specific functionalities into individual network nodes via mesh models. When all the necessary components are in place, a complex sensor-driven lighting system can then be built.

By defining basic functionalities of network nodes, *mesh models* address the problem of interoperability at the application level. A model defines the basic functionality of a node. *Server models* define states, supported messages, and state transition behavior. *Client models* define a set of messages that a client can use to communicate with a server. A *control model* may also contain control logic to coordinate interactions between other models. *Foundation models* define the access layer states, messages, and models required to configure and manage a mesh network. To increase design flexibility, models include multiple properties that can be adjusted in accordance with specific applications. Simple devices may rely on very few models, but more complex devices need to employ many models—including certain models that require that other models be instantiated within the same node; this is known as *model extending*.

To function properly within a wireless mesh network, each device implements two Foundation Models:

- Configuration Server model (used for network management)
- Health Server model (used to represent mesh network diagnostics of a device)

These Foundation Models are defined in the Mesh Profile Specification [2]; all of the other models referred to in this white paper are defined in the Mesh Model Specification [1].

Sections 2-10 introduce and discuss the system architecture and each component of the adaptive lighting system. Section 11 then discusses the complete installation.

This white paper assumes that the reader is familiar with the basics of Bluetooth mesh networking. An overview of the most important concepts and terms can be found in the Mesh Profile Specification [2], Section 2.3.

2 System architecture

Lighting control systems consist of one or several line-powered light sources that can be controlled via different mechanisms.

Light sources (e.g., LED modules) typically are combined with electrical components (e.g., transformers and drivers), optical components (e.g., reflectors, diffusors, and lenses), and mechanical components (e.g., housing) that are required to provide a ready-to-use product. Such a product is commonly called a *luminaire* and forms the basic component of a lighting control network. Each scenario described in this white paper assumes that at least one luminaire is present.

Luminaires can be controlled by switches, sensors, smartphones and tablet personal computers (PCs), or components of building automation systems:

Switches allow manual adjustment of the light level. Users can switch the light on or off, increase (dim up) or decrease (dim down) the light level, or recall specific light levels that were defined (“lighting scenes”) based on their preferences.

Sensors allow automatic adjustment of the light level based on detected parameters such as room occupancy and available light (e.g., daylight coming through windows) based on defined rules.

Smartphones and tablet PCs can be used both to control the light level (instead of using switches) and to configure the lighting control system.

Lighting control systems might also be integrated into larger-scale **building automation systems**, which control and monitor various functions of a building (e.g., lighting, air conditioning, hazard monitoring, and access control). Building automation systems might both monitor the status of the lighting control system (e.g., room occupancy) and request specific settings (e.g., turning on all lights in case of a fire alarm).

Figure 2.1 shows an example of a mesh network for one such lighting control system, which consists of the following components:

- Luminaires (represented by circles with an “X” symbol inside)
- Sensors (occupancy, ambient light)
- A wall switch (also often referred to as a wall station)
- A smartphone or a tablet (used to configure the system and interact with it)
- A Building Management System (BMS)

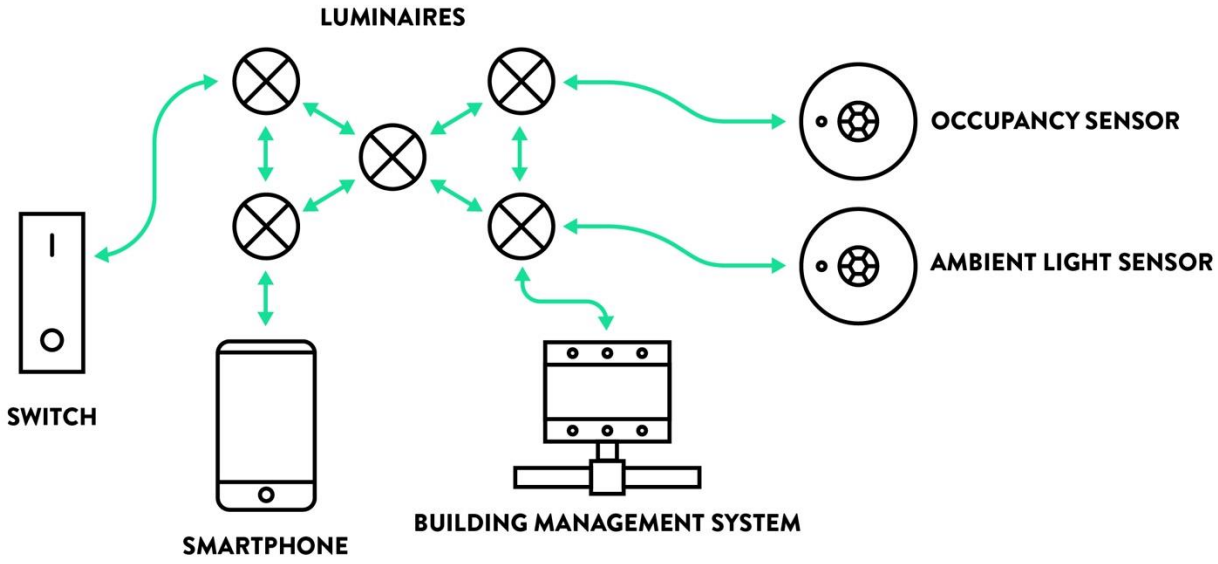

Figure 2.1: Example of a lighting control network

3 Switches

As noted earlier, luminaires are the basic network component of a lighting control system. In traditional lighting systems, luminaires are typically controlled using wall switches. Although wireless networks enable more ways to control lights, wall switches remain the primary tool for manual lighting control.

With the ultra-low-power characteristics of the Bluetooth radio, as described in the Bluetooth Core Specification [3] Volume 2, Part A, a lighting control system based on Bluetooth mesh enables using wireless switches. A wireless switch enables more flexible configuration and operation than does a traditional wired switch, yet both types of switch can be used in almost exactly the same way. There are some slight differences in how wireless switches communicate with luminaires, which is a consequence of how wireless systems work in general. This section describes these differences.

3.1 Bi-stable on/off switches

On/off switches are the most basic form of controlling lights. On/off switches allow the light output of a luminaire, at a minimum, to be switched either on or off. This functionality has existed since the beginning of electrical lighting, and many users continue to expect this type of light control.

Initially, most light switches were designed as *bi-stable on/off switches*, meaning they could be set to one of two possible positions, reflecting either the on state or the off state. Figure 3.1 shows an example of such a bi-stable on/off switch.

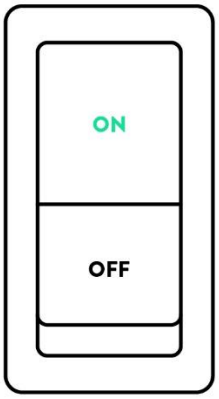

Figure 3.1: Bi-stable on/off switch

Bi-stable on/off switches are easy and intuitive to use as long as only one switch controls a luminaire. In many scenarios, however, it is desirable to have more than one switch per luminaire—for example, to allow luminaire control from both the door and the bed in a hotel room. The occupant typically might use the switch by the door to turn the lights on when entering the room and the switch by the bed to turn the light off. This means, however, that the switch by the door remains in the “on” position when the light is switched off using the switch near the bed. For this reason, lighting systems based on Bluetooth mesh do not use bi-stable on/off switches. This limitation can be overcome either by using toggle switches or by using momentary switches, described in Section 3.2 and Section 3.3, respectively.

3.2 Toggle switches

Toggle switches enable alternating the light output of a connected luminaire between two states whenever one of the switches changes state—typically, between an on state and an off state.

The most common implementation for toggle switches is to use a set of two single-pole, double-throw (SPDT) switches, as shown in Figure 3.2. This figure shows two such switches (Switch 1 and Switch 2) where each of these switches connects one input with one of its two outputs depending on the switch position (indicated by the green arrow).

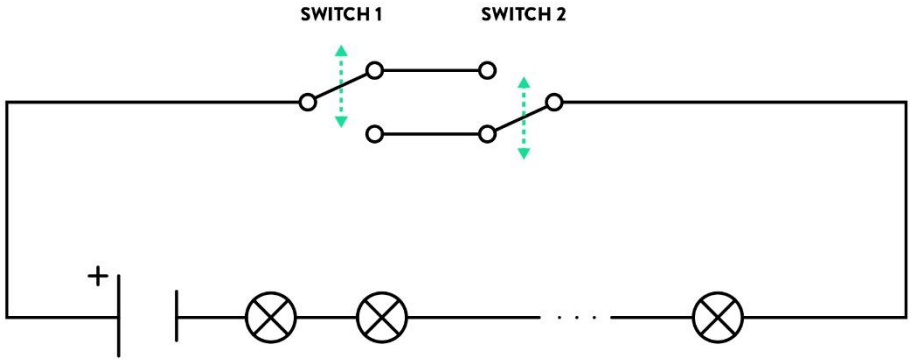

Figure 3.2: Toggle switch implementation for two switches

The design shown above can be generalized to support more than two switches by using double-pole, double-throw (DPDT) switches and connecting them. One possible implementation for more than two switches is shown in Figure 3.3. Note that the first (leftmost) and the last (rightmost) switches in this chain could also be implemented as double-pole, single-throw (DPST) switches.

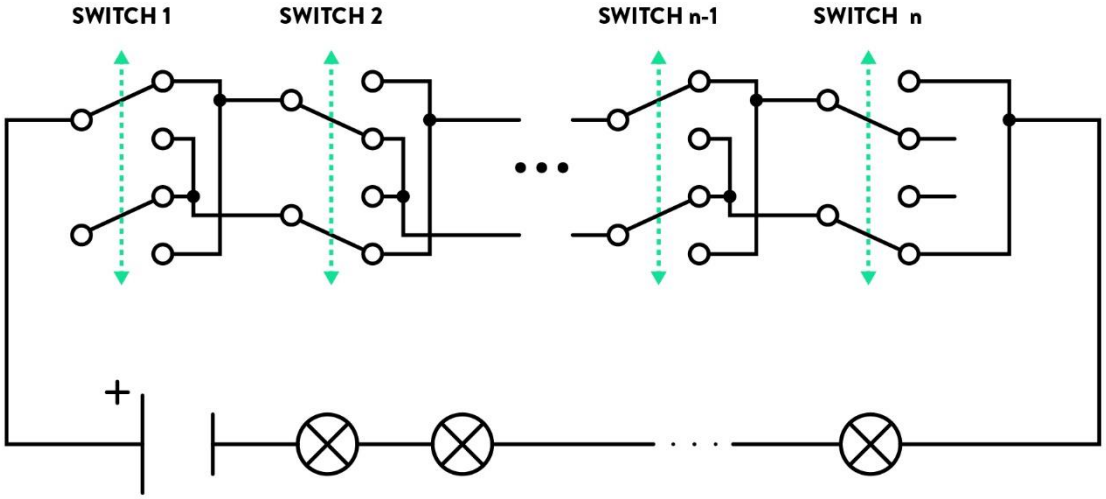

Figure 3.3: Toggle switch implementation for more than two switches

Based on these examples, you can identify two key properties of lighting control implementations that use toggle switches:

- The reaction to a user's input (flicking the switch) depends on the current state of the system. If the lights are on, flicking a switch turns them off. If the lights are off, flicking a switch turns them on.
- The connected luminaires are expected to have the same state. Either all luminaires are on or all luminaires are off.

Both properties pose challenges within modern wireless lighting control scenarios—both because of the need for idempotency within wireless networks (see Section 3.2.1) and synchronization issues (see Section 3.2.2).

3.2.1 Idempotency

Wireless networks can usually provide quite reliable communication, as measured by the percentage of correctly received messages. However, it is not possible to guarantee that all messages will be received. Limiting factors can be both external (e.g., interference from other radio systems) and internal (e.g., very high network load).

Wireless networks can address this challenge by repeatedly sending the same message until all intended receivers have received it. As a result, each receiver might receive the same message a different number of times. The state of each receiver (e.g., light being on or off) should in these cases be independent of the number of messages the receiver receives.

One design principle meeting this requirement is *idempotency*, a concept used both in mathematics and computer science. This states that the result of an operation remains the same, no matter how often that operation is executed. Taking an example from mathematics, the output of the function f(x) = x remains the same no matter how often this equation is executed.

To help avoid potential side-effects of a receiver receiving the same message several times, Bluetooth mesh is designed to be an idempotent network.

In a lighting control system, many messages, such as On, Off, Scene Store, or Scene Recall, are idempotent. The state of the light source that receives one of these messages remains the same no matter how often the light source receives the message.

A toggle message, however, is not idempotent: the receiver of such a message alternates between its two supported states each time it receives the message. Designing a lighting control system that resolves these issues with toggle functionality, as well as the synchronization issues discussed in Section 3.2.2, requires specific implementation mechanisms.

3.2.2 Synchronization

As noted earlier, one key property of a lighting control system that uses toggle switches is that connected luminaires are usually expected to have the same state (on or off) at any given time. Often, several luminaires are grouped together, as shown in Figure 3.2, and are switched wirelessly.

One classic example is a chandelier with several Bluetooth mesh-controlled bulbs that are configured as a group. If a toggle switch is used to control the bulbs, the expectation is that if all lights in the group are on, a toggle message would switch all the lights off. Conversely, if all lights in the group are off, a toggle message would be expected to turn them on.

This seemingly straightforward logic, however, leaves one important question unanswered: What happens if all of the lights within the group do not have the same state?

Real-life experience with wireless systems shows that because of various factors, there is no guarantee that all lights in a group will always have the same state. For example, some luminaires within the group might still be connected to legacy light switches, and power might be cut temporarily by users turning off the legacy light switch on the wall. Alternatively, if some lights within the group are physically located at the edge of usable radio communication distance, these lights might miss one or several messages because of signal degradation or radio interference.

If a toggle message is sent under these conditions, then some of the lights within the group might not receive it successfully. To illustrate this problem, consider the system shown in Figure 2.1. and a scenario in which the switch sends a toggle message to the group address representing all five lights to switch the

lights on. The three lights closest to the switch might receive such a message. However, the two lights farther away might not receive such a message because another message is being sent at the same time by a node in close proximity.

The three lights that received the message would change their state (to on) while the two lights that didn't would retain their previous state (off). Clearly, this is not the intended result. To address the problem, the switch could, of course, send the message again so that the remaining two lights also receive the message and transition their state. If this succeeded, the two luminaries that had remained off would now be turned on, but the three luminaires that successfully turned on previously would now be turned off.

Obviously, repeating a toggle message would not resolve the issue of lights getting out of synchronization. This is a consequence of a toggle message not being idempotent.

Once the luminaires are out of synchronization, the only practical way to resynchronize all lights is to issue an idempotent message—such as Switch On—to all of them. This could be done, even in the case of toggle switches, by using a dedicated trigger. For example, the user could press and hold the switch for 5 seconds.

However, in real life, a user would not expect this, and would probably not remember how to resolve this issue. They might have to resort to consulting an operating manual every time the lights get out of sync. A user who is not tech-savvy might see this as "fixing a bug" in the lighting system and lose confidence in it.

Because of the issues with idempotency and synchronization, many implementations of wireless lighting control systems, including systems based on Bluetooth mesh, do not use non-idempotent messages such as the toggle message described here.

3.2.3 Design guidelines for achieving toggle functionality

It is possible, however, to achieve the desired toggle functionality by using On/Off messages and one of the following approaches:

- The toggle switch or sensor subscribes to the same group address as the luminaires (implementing a relevant server model such as the Generic OnOff Server) and is therefore aware of the current on/off state of the group. When sensor or user input is received, the toggle switch or sensor checks the current state of the luminaires and accordingly generates either an On or Off message.

  Limitation: For battery-operated switches, this might not be practical because of the amount of power required to monitor the state of the group.

- The toggle switch or sensor alternates between sending On and Off messages. For example, the switch sends the On message on a first press, the Off message on a second press, the On message again on a third press, and so on.

  Limitation: With this method, the user might need to trigger the switch or sensor twice (especially if more than one switch is used to control the lights), which would, again, be seen as a flaw in the wireless lighting control system.

- The toggle switch or sensor requests the current state from one member of the group of luminaires and generates a message changing the group's state in respect to the status of this member.

  Limitations: This method introduces latency that may be too long to provide good visual user feedback. Also, there is a risk that the reference node could be unreachable because it is out of transmission range or simply powered down. Finally, the reference node could be out of sync with other members of the group.

- The toggle switch or sensor requests the current status from all members of the group and generates an On or Off message based on a majority vote.

  Limitation: This method probably would provide the closest equivalent to the desired function, but the time required to interrogate all group members would usually be unacceptable.

Of these options, only the first approach seems practical. The remaining options might be feasible in certain use cases—for example, for a sensor for which the latency between a trigger and the toggle event is irrelevant.

Synchronization issues can be avoided if switches are implemented as momentary. That is, the circuit is continuously opened or closed when force is applied to the switch; when force is removed, the switch returns to its normal state. This implementation has become standard in many wireless lighting systems and is described in Section 3.3.

3.3 Momentary switches

Momentary switches provide dedicated on and off positions, much like the bi-stable on/off switches discussed in Section 3.1. The key difference between momentary switches and bi-stable switches is that momentary switches do not maintain the position to which they are actuated. Instead, they are "spring return" switches that automatically return to their original non-momentary (neutral) position as soon as the user lets go of them. Momentary switches send a message as soon as they are pressed; how long the user presses the switch is not relevant.

The neutral position of a momentary switch does not represent any state. Therefore, the state of the switch is never in conflict with the state of the luminaires when their state is changed, even if the state change of the luminaires is caused by another switch or as a result of an automatic action initiated by a timer or an occupancy sensor.

Figure 3.4 illustrates a sample implementation of a momentary switch. In the first view, the switch is in a neutral position. When the user presses the on button (labeled “1”), an On message is sent; the physical switch then returns to the neutral position. When the user presses the off button (labeled “0”), an Off message is sent, and the physical switch again returns to the neutral position.

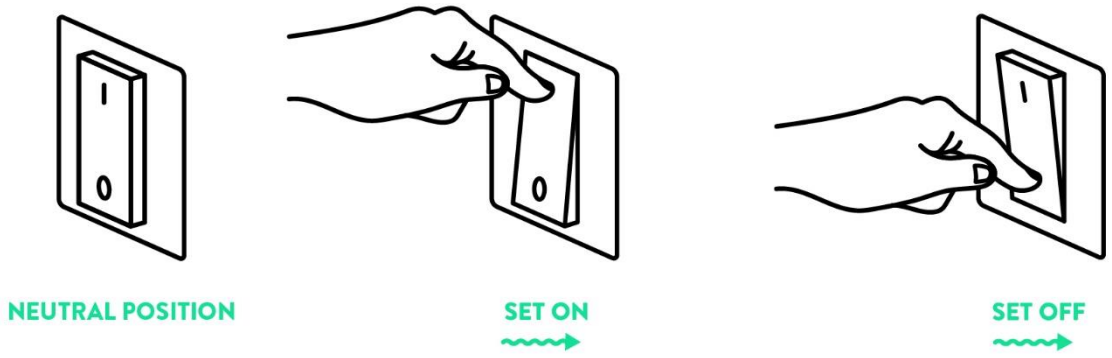

Figure 3.4: A momentary switch for a luminaire in action

The most basic lighting control scenario involves a luminaire and an on/off switch. To enable this simple system on a Bluetooth mesh network, the luminaire implements the Generic OnOff Server model, and the switch implements the Generic OnOff Client model. These are generic models, which means their

functionality is non-specific. They can be used in a variety of applications, to turn various devices, including lighting, on and off.

Within this scenario, the on/off switch (implementing a client model) could control the luminaire (implementing a server model) by *pusblishing* its state (on or off) to the luminaire which *subscribes* to these publications. The luminaire would then react to the switch state publications by changing its own state, e.g. turning the light on or off. The luminaire itself does not require any specific information about the switch. This approach therefore allows building a simple and straightforward lighting control system involving one on/off switch and one luminaire.

While wireless switches implementing Bluetooth mesh networking can be operated similarly to traditional wired switches, there are some fundamental differences in the way the wireless switches interact with luminaires.

The simplified diagram in Figure 3.5 illustrates how a traditional analog on/off switch works. In wired systems, an on/off switch has a state, which can be open (“off”) or closed (“on”). Whenever a switch is pressed, its state changes. Pressing “off” opens the circuit, interrupting the flow of power. Pressing “on” closes the circuit, letting the power flow through the switch to the luminaire.

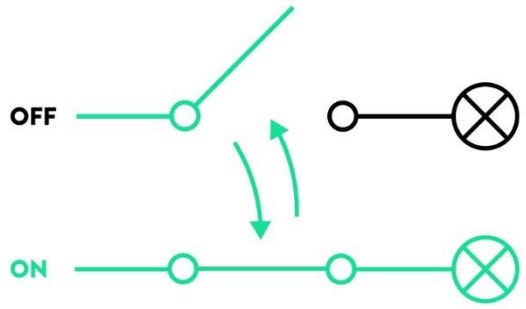

Figure 3.5: Analog (wired) on/off switch operation

In contrast, Figure 3.6 illustrates the operation of a digital wireless on/off switch (shown on the left) controlling the state of a luminaire (shown on the right). In a Bluetooth mesh network, an on/off switch does not have a state. Pressing a momentary switch triggers a relevant message (Set On/Set Off), which sets the Generic OnOff state of a luminaire. If the message is an *acknowledged message* (i.e., the message requires a response), a corresponding status acknowledgment message is sent back to the originator. An *unacknowledged message* will not trigger any response from the luminaire.

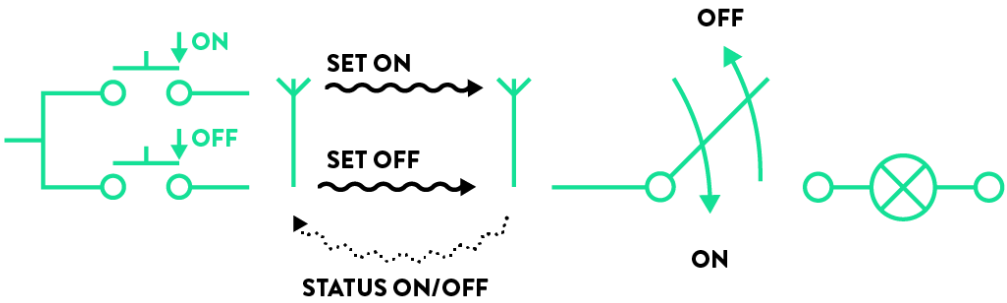

Figure 3.6: Digital (wireless) on/off switch operation

In commercial settings, switches are rarely used to control a single luminaire. To control a group of luminaires by using a single switch, a group address needs to be created and set as the model’s publish

address. As in the previous example, this information must be specified for the Generic OnOff Client model that is used by the switch. In addition, the luminaires that are controlled in this way must act upon messages published to the entire group.

To do this, the Generic OnOff Server model that is implemented by each of the luminaires must be subscribed to the group address—the same address that has been specified as the Publish Address for the switch. Figure 3.7 shows a group of luminaires and the address configuration for an on/off switch that controls all of them simultaneously.

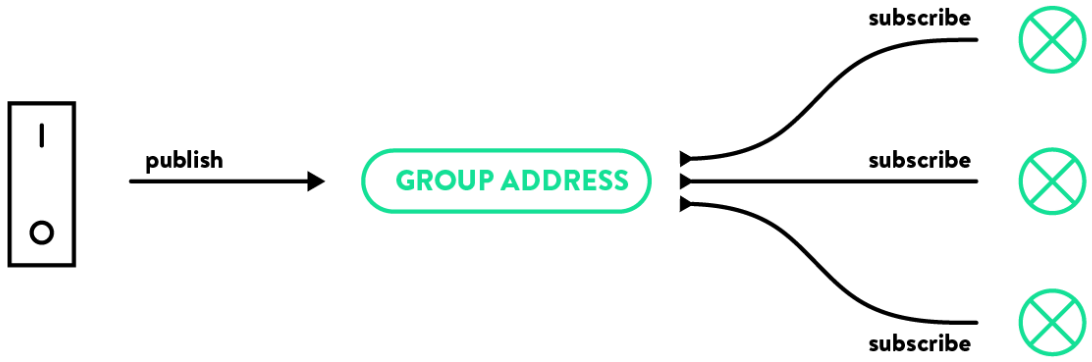

Figure 3.7: A single switch that controls multiple luminaires

Client models implemented by switches only send messages: they do not themselves have states. This is why adding additional switches to the network will not create any conflicts. Each additional switch will, again, need to implement the Generic OnOff Client model with the same group address that has been specified for other switches (as their Publish Address) and corresponding luminaires (as the group address to which they are subscribed). This way, each additional switch can control the same group of lights independently. Figure 3.8 illustrates this concept by showing two switches controlling the same group of lights.

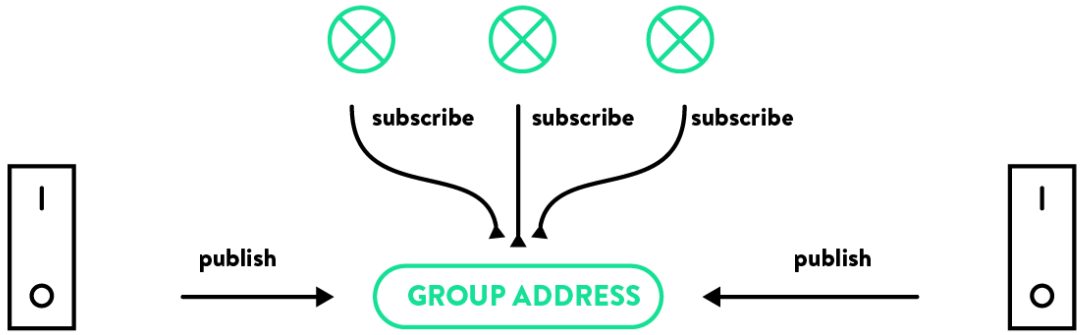

Figure 3.8: Two switches that independently control a group of luminaires

Finally, another scenario would be to have multiple switches controlling multiple luminaires that are assigned to multiple overlapping groups. A practical example of such an arrangement could be an office floor with one main switch that controls all of the luminaires and several switches that control luminaires within individual rooms. Figure 3.9 illustrates a simple example of such a system with one main switch (shown on the left) controlling all three luminaires and another switch (shown on the right) controlling only two luminaires.

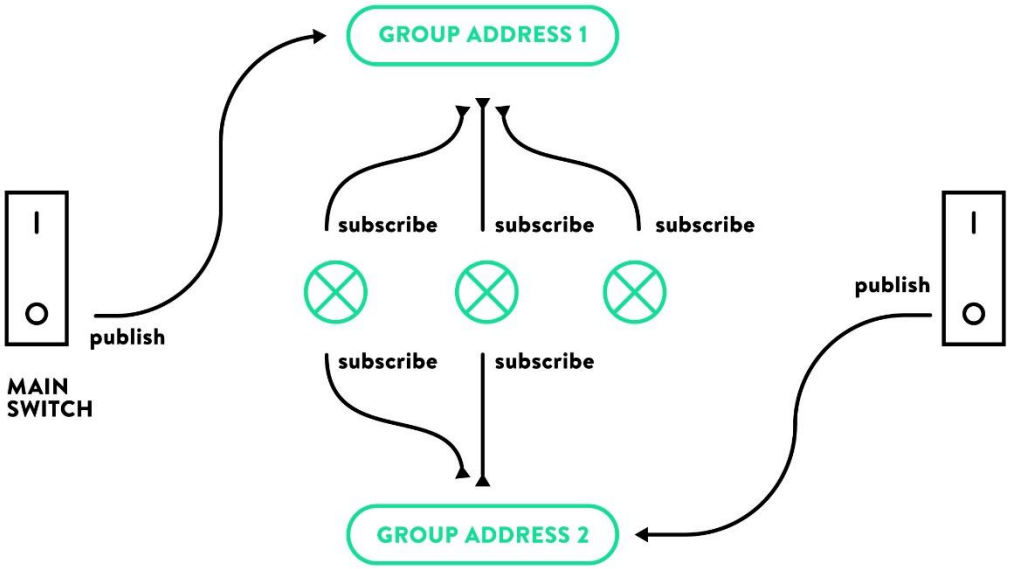

Figure 3.9: Overlapping groups of luminaires controlled by multiple switches

This system can be built using the same methods and requirements used to build the previous systems. No additional mesh models are required, and the example system has three luminaires and two switches. One of the switches controls all of the luminaires while another switch controls only two of them. This is accomplished by creating two overlapping groups, which requires configuring two different group addresses. One of the luminaires is subscribed to *group address 1* only; the remaining luminaires are subscribed to both group addresses.

So far, we've built several simple networks with on/off switches controlling multiple luminaires. We've also learned the concept of node grouping. The principles used for grouping can be applied to a number of other applications defined in the Mesh Model Specification [1]. The majority of scenarios presented in the rest of this paper can be implemented using the configurations described earlier. Using these simple examples, we've also learned the basics of the client-server architecture communicating via a *publish-subscribe paradigm*, one of the basic concepts in Bluetooth mesh networking. The publish-subscribe communication model moves the focus from nodes to information, resembling the way some social networks and interest tags work. Addresses may be assigned to information, not only to specific devices. Luminaires subscribe to these addresses so that they can respond to messages that concern them. Other devices, such as sensors and switches, publish to these addresses, distributing lighting control messages among all interested parties at the same time.

3.4 Achieving synchronization using the Delay field

One of the major challenges in the wireless lighting control application is the synchronous operation of luminaires. Because of the "lossy" nature of the wireless medium, whenever a message (e.g., a Generic OnOff Set message setting the lights to "on") is sent to multiple luminaires, chances are that some destinations will fail to receive it.

This is addressed in Bluetooth mesh networking by configuring senders to transmit messages multiple times. Three transmissions typically give a good balance between reliability and network performance. This number is high enough to significantly increase the reliability of message delivery but at the same time low enough to help minimize the impact on network congestion.

How effectively does this solve the problem of message loss? Let's assume that approximately 90 percent of messages are delivered successfully in a given wireless environment. This means that for a single message, the probability of failed delivery amounts to 10 percent. But if three separate delivery attempts are made for this message, the probability that all of them fail amounts to only $(10\%)^3$, which equals 0.1 percent. Therefore, the reliability goes up from 90 percent to 99.9 percent.

Retransmitting messages can be a very effective method for improving reliability of mesh networks, but this method does have its price in the demanding lighting environment. Imagine a ceiling with dozens of luminaires operating as a single group. A switch is pressed, sending the "on" message to all of them. This message is then transmitted two more times as described above. Roughly 90 percent of luminaires receive the first message and act upon it (i.e., they turn on). But the remaining luminaires turn on only after they receive the second copy of the message. And one or two luminaires might need a third repetition. The time that passes between the delivery of these three messages is very short and can be counted in milliseconds. But the human eye will notice these differences very clearly. The group of lights will turn on asynchronously, showing what is called the *popcorn effect*. There are relevant mechanisms in Bluetooth mesh networking that can address this issue.

Messages may include a Delay field that indicates a duration of time the server waits before executing the state-changing model behaviors for a particular message. This message execution delay helps synchronize actions of multiple receivers (e.g., luminaires) when senders retransmit messages multiple times. The Delay field can be different for each retransmitted message. This way, it is possible to compensate for the time elapsed since transmitting the first message. Figure 3.10 illustrates how the Delay field enables synchronous operation of a group of luminaires. In this scenario, we assume that all luminaires within the group can receive signals from the switch and that we need to use retransmission due to packet loss to ensure that all lights have received the message at least once.

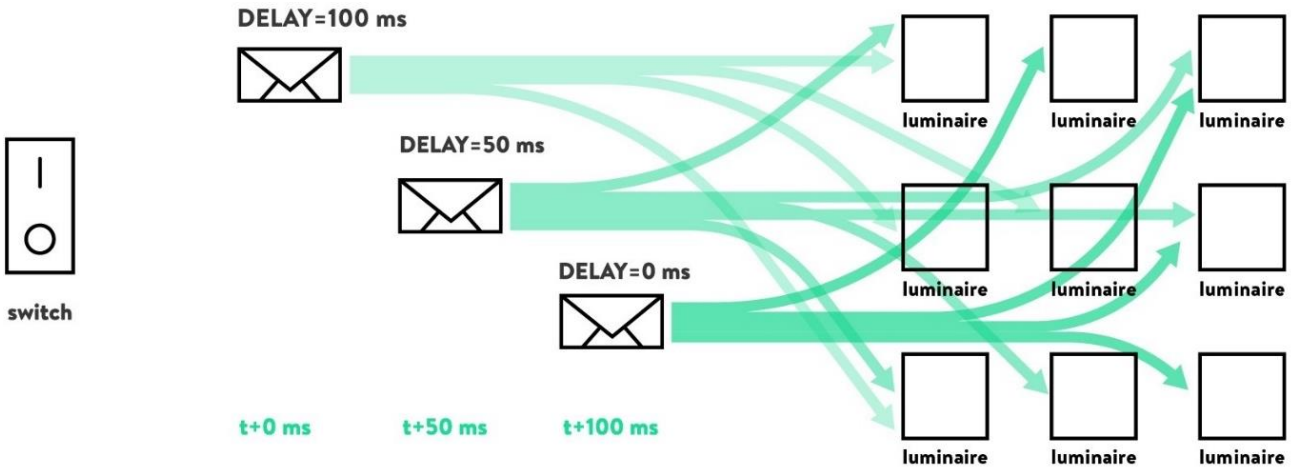

Figure 3.10: Using the Delay field to minimize the popcorn effect

The first message is sent to the luminaires immediately (+0 ms) after the switch is pressed. The Delay field of the first message is set to 100 ms. After 50 ms, the second message is sent, this time with the Delay field set to 50 ms. Finally, 100 ms after the switch was pressed, the third message is sent with the Delay field set to 0 ms.

Even though these three messages are delivered to the luminaires with 50 ms intervals between them, they will all be executed at the same time as a result of the Delay field configuration. From the end user perspective, the luminaires will turn on fully synchronously after the switch is pressed. Yes, there will be a

slight delay in the luminaires' response, but the perception of the delay is far less noticeable than the lack of synchronicity.

The numbers above are not arbitrary; a total delay of 100 ms between pressing the switch and having the luminaires turn on cannot be noticed by the human eye. What is important is that the luminaires all turn on at once. If there were similar differences between individual luminaires in the group, a person would clearly see their operation as asynchronous.

3.5 Summary of on/off control scenario

The description in this chapter shows how Bluetooth mesh networking technology enables simple on/off lighting control. Table 3.1 below summarizes the configuration for the involved nodes.

Nodes Involved	Luminaires	On/off switches
Models Used	Generic OnOff Server	Generic OnOff Client
Publish / Subscribe	Subscribe	Publish

Table 3.1: Configuration summary for a wireless on/off switch

Figure 3.11 shows the relationship between the models used in this scenario.

Figure 3.11: Relationships between the models described in this section

4 Light level control

To make our lighting control system more flexible, we'll add light level control capabilities, including power-up behavior, dimming, and real-time light level controls.

4.1 Power-up behavior

The Mesh Model Specification [1] allows you to decide what happens when a luminaire is powered up—i.e., when power is physically supplied to a server device following a power outage. To enable power-up behavior to be specified, a luminaire requires the Generic Power OnOff Server model and the Generic Power OnOff Setup Server model. The Generic Power OnOff Server model extends the Generic OnOff Server model, so the Generic OnOff Server model also needs to be implemented. The power-up behavior is determined by the Generic OnPowerUp state, which can be set to one of the following modes:

- **Off:** When a luminaire is powered up, its light level is set to 0 percent.
- **Default:** When a luminaire is powered up, its light level is set to a desired default value.
- **Restore:** When a luminaire is powered up, it restores the same light level it had when the power outage occurred.

  Note: The reason for defining two separate models (Generic Power OnOff Server and Generic Power OnOff Setup Server) is to enable the configuration of access-level security as described in the Mesh Profile Specification [2]. In Bluetooth mesh, access-level security is provided per model. So the Generic Power OnOff Setup Server model can be configured to require a different application key compared to the application key that the Generic Power OnOff Server model requires. The Generic Power OnOff Setup Server model may be configured to require a system installer's key to change the configuration settings (e.g. the default light level after power up), while the Generic Power OnOff Server model may be configured to require just a standard user's key to turn the device on and off.

4.2 Dimming

The Mesh Model Specification [1] supports real-time light level control capabilities, which enable the familiar dimming functionality available on wired systems. To enable light output adjustments, you need to implement the Generic Level Server model and the Generic Level Client model. As in the previous examples, the server model is implemented in the luminaires, and the dimmers rely on the client model. This follows a general rule that all lighting control scenarios in Bluetooth mesh use.

The generic level assumes a linear relationship between the level input and the perceived output. However, this does not work well for lighting because of the nature of the human eye. To achieve perceived linear dimming, the intensity of light must be matched to the way our eyes work.

The Light Lightness Server model (which extends the Generic Power OnOff Server model and the Generic Level Server model) and the Light Lightness Client model take care of this. They introduce the Light Lightness Linear state and the Light Lightness Actual state. The former represents the lightness on a linear scale; the latter represents the lightness on a perceptually uniform lightness scale. The Light Lightness Linear state is bound to the Light Lightness Actual state, which in turn is bound to the Generic Level state.

This arrangement helps to provide a smooth, dimming experience that is friendly to the human eye. Because the Light Lightness Linear state and the Light Lightness Actual state are *bound*, whenever one of the states changes, the other state changes accordingly. This change is implemented in accordance

with a logarithmic dimming curve so that the perceived lightness of a light is approximately the square root of the measured luminous intensity. These relationships are explained in the Mesh Model Specification [1], Appendix A.2.

A dimmer can be used to control the light level if it implements the Light Lightness Client model (often in conjunction with the Generic Power OnOff Client model used to control the initial light level at power up). A dimmer can also be used to control the light level if it implements the Generic Level Client model. The dimming operation will be the same because of the bi-directional binding between the Light Lightness state and the Generic Level state of a luminaire. However, if a device implements the Light Lightness Client model, the device can only control the light level.

The Generic Level Client model is a bit more universal. For example, this model can be used to control the color temperature of a luminaire (often referred to as *tunable white*) or a level of a non-lighting device, such as a window blind. On the server side, both the Generic Level Server model and the Light Lightness Server model are required because the Light Lightness Server model extends the Generic Level Server model and the Generic Power OnOff Server model.

In a Bluetooth mesh lighting network, dimming can be realized in two ways:

- The light level can be adjusted to reach a desired value within the range 0 percent to 100 percent. The desired value is a given absolute value, regardless of the previous light level. Models involved in this operation interact with each other through two states: the Light Lightness state and the Generic Level state. These states are bound *bi-directionally*, which means that whenever one of these states is changed, the other adjusts accordingly. Therefore, a desired light level can be set by a client that controls either the Generic Level state (through Generic Level Set messages) or the Light Lightness state (through Light Lightness Set messages).
- The light level can be adjusted by a specified Delta value (e.g., +15 percent). This results in an incremental change in light output. The process supports changing the state by a cumulative value by using a sequence of messages that are part of a transaction. This means that a dimmer can be used to perform multiple Delta adjustments, which will be sequenced into one cumulative operation. Figure 4.1 shows how three such Delta adjustments are sequenced into one cumulative operation.

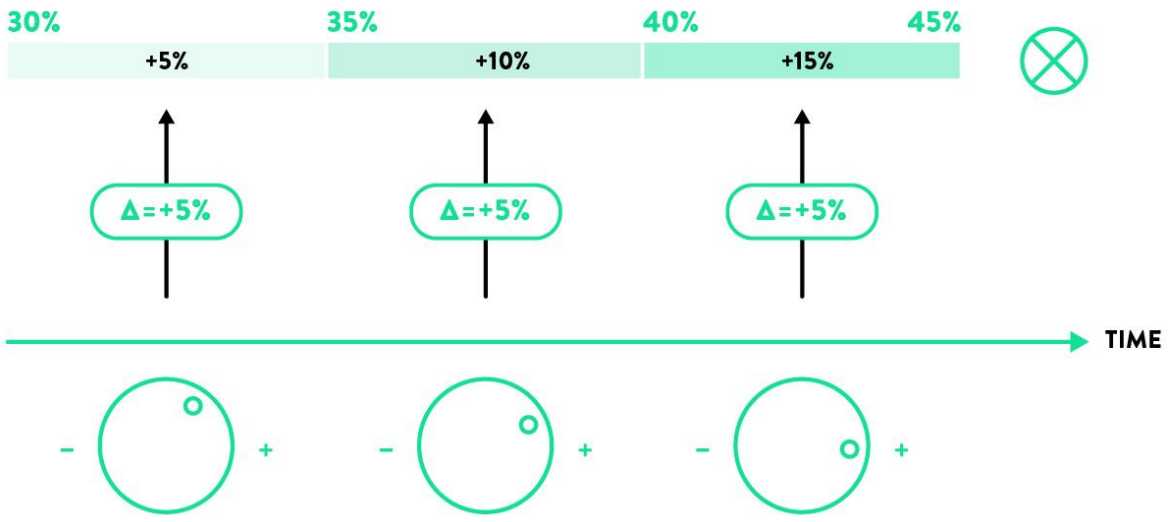

Figure 4.1: Incremental dimming: multiple Delta adjustments sequenced into one cumulative operation

4.3 Dimming adjustment through transactions

The concept of *transactions*—which are used to adjust the light level by a specified Delta value—is one of many innovative solutions introduced in Bluetooth mesh networking to address the challenging requirements of the lighting environment.

What does the transactional nature of dimming mean in practice? Each dimmer rotation operation has its own transaction identifier (TID). When a new transaction identifier is generated, the luminaire receives a Delta Set message with a transaction ID attached to it, and the luminaire remembers its latest light level as a new base level. Each message received as part of a given transaction will relate to this particular base level, increasing the light output of a luminaire by a specified Delta value. The messages within a transaction carry the cumulative values of the Delta Level field. As a result, if a radio packet collision occurs, and certain messages within a transaction are not received, the next received message makes up for the lost messages, carrying cumulative Delta values.

The transaction is closed when no messages arrive from the source address for a period of 6 seconds—i.e., when a dimmer is not rotated for a period of 6 seconds. A new transaction is opened when the transaction ID in the received message is different from the transaction ID in the previously received message that used the same source and destination addresses.

4.4 Summary of light level control scenario

The description in this chapter shows how Bluetooth mesh networking technology enables light level control with dim functionality. Table 4.1 below summarizes the configuration for the involved nodes.

Nodes Involved	Luminaires	Dimmers
Models Used	Generic OnOff Server	Generic OnOff Client
	Generic Power OnOff Server	–
	Generic Power OnOff Setup Server	–
	Generic Level Server	Generic Level Client
	Light Lightness Server	Light Lightness Client
	Light Lightness Setup Server	
Publish / Subscribe	Subscribe	Publish

Table 4.1: Configuration summary for a light level control

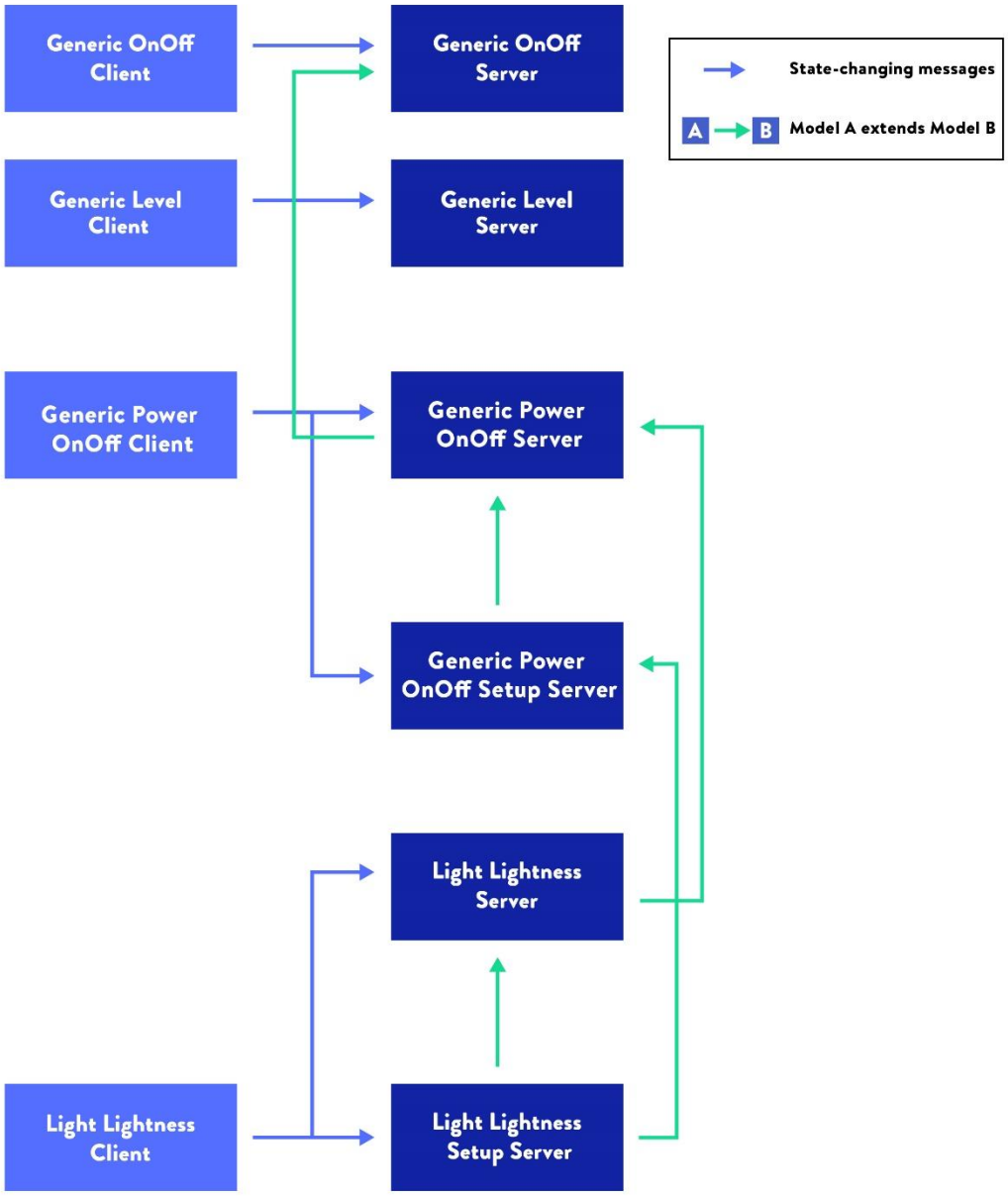

Figure 4.2: Relationships between the models described in this section

5 Transitions

In Section 4, we learned how to control the light level using a dimmer, but not how to control the time it takes to transition the light level. Whenever a dimmer requests that a luminaire change its light level from one value to another (e.g., when a new absolute light level value is requested), the question of how long this process should take naturally arises. The time it takes for a state to change from its present state to the target state is called the *transition time*.

The transition time can be adjusted flexibly using the Generic Default Transition Time Server model, as described in the Mesh Model Specification [1]. That model covers a wide range of time increments, from milliseconds to hours, and is therefore capable of handling a variety of applications.

The Generic Default Transition Time Server model imposes the following procedures:

- If the transition time is not specified in a state-changing message sent by a client, then use the default transition time specified for this device. The default transition time can be changed by the client using a Generic Default Transition Time Set message.
- If the transition time is specified in a state-changing message sent by a client, apply it.

Figure 5.1 illustrates how transition time affects the transition of a luminaire that has been instructed to turn on (100 percent output). After the message arrives, the luminaire increases its light level from 0 percent to 100 percent over the specified transition time (TT).

Figure 5.1: The transition time can be used to determine how long it takes a luminaire to increase its output from 0 percent to 100 percent

5.1 Summary of transition functionality

The description above shows how Bluetooth mesh networking technology enables the light to transition between different states. Table 5.1 below summarizes the configuration for the involved nodes.

Nodes Involved	Luminaires	Configuration devices
Models Used	Generic Default Transition Time Server	Generic Default Transition Time Client

Table 5.1: Configuration summary for light level transitions

Figure 5.2: Relationships between the models described in this section

6 Scenes

The Mesh Model Specification [1] introduces a mechanism called scenes that allows for memorizing the states of devices (e.g., light level or color) so that the states can be restored on demand or on a pre-set schedule. In a luminaire, scenes are handled by two models. The Scene Server and the Scene Setup Server models take care of scene configuration, making it possible to memorize desired states of devices. The Scene Server model is used to restore the previously saved scenes. On the switch, scenes are handled by the Scene Client model.

Note: As explained earlier, the reason for defining two separate models (Scene Server model and Scene Setup Server model) is to enable the configuration of access-level security. You can configure the Scene Setup Server model to require a system installer's key to change scene settings. On the other hand, you can configure the Scene Server model to require just a standard user's key to recall a desired scene.

Let's say you want to create a scene involving three luminaires, each with a different light level: 10 percent, 20 percent, and 30 percent. To set up the scene for the first time, the desired light levels need to be set for each luminaire, one by one, using the methods described in Section 4. Then, the Scene Store operation needs to be performed for each of these luminaires. As part of this process, a message is sent to the Scene Setup Server model in each luminaire, requesting that this model memorizes the state of the luminaire under a selected Scene Number. Because the Scene Number is a 16-bit, network-wide value, a mesh network can accommodate as many as 65,535 scenes. If a luminaire is transitioning when the Scene Store message arrives, then the luminaire's target light level will be stored for that particular scene. Figure 6.1 illustrates the process of storing the light level for a scene.

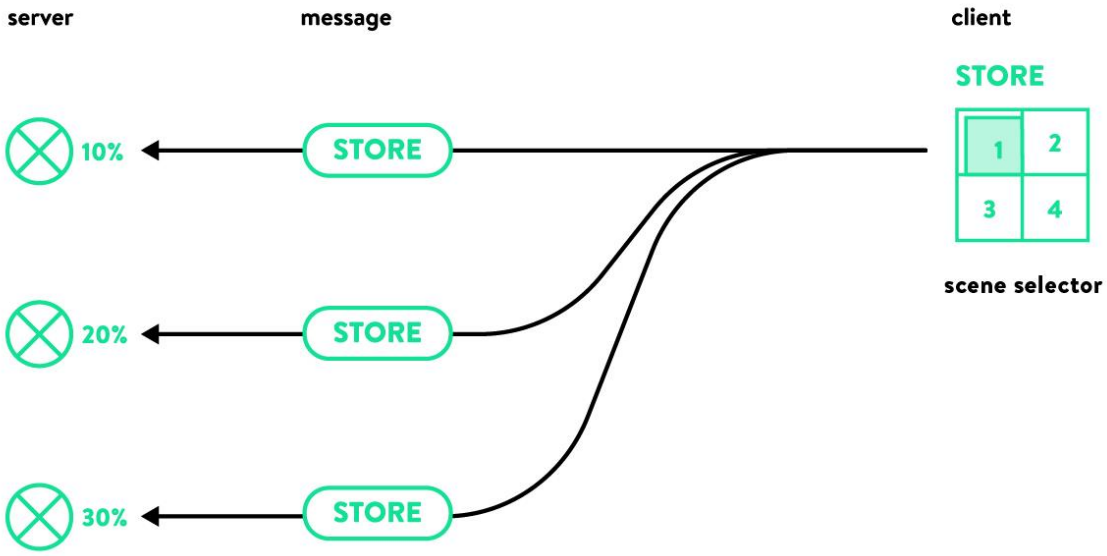

Figure 6.1: The Scene Store operation

After the Scene Store operation is performed for all three of the luminaires, you can recall the scene whenever you need to by referencing the Scene Number. The recall operation, called Scene Recall, applies the stored values of states to all luminaires included in a given scene. If the Scene Recall message arrives while a luminaire is in transition, the transition process is interrupted, and a new transition time period is initiated. In this example, the Scene Recall operation sets the light level of the luminaires to 10 percent, 20 percent, and 30 percent, respectively, regardless of their previous states. Figure 6.2 illustrates how the lights transition to the target light level during the Scene Recall operation.

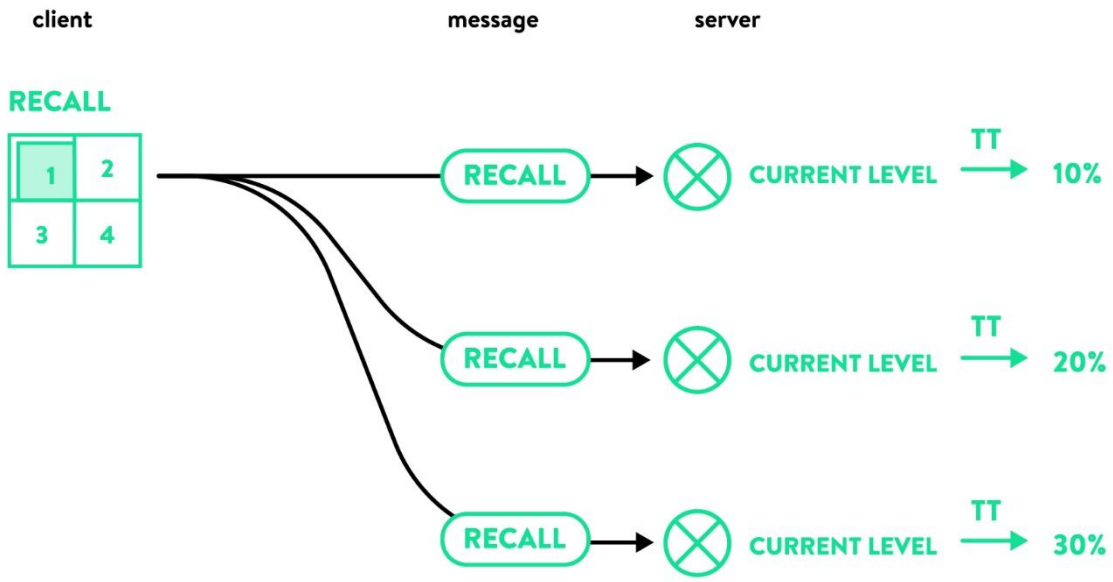

Figure 6.2: The Scene Recall operation

The Scene Recall message can specify the transition time for a given scene, which will make all of the luminaires reach their previously saved light levels over the same period of time.

6.1 Summary of scene functionality

The description above outlines how to use scenes to define (set) and use (recall) light levels for one or several luminaires. Table 6.1 below summarizes the configuration for the involved nodes.

Nodes Involved	Luminaires	Switches/Dimmers	Configuration Devices
Models Used	Scene Setup Server	–	Scene Client
	Scene Server	Scene Client	
Publish / Subscribe	subscribe	Publish	–

Table 6.1: Configuration summary for scenes

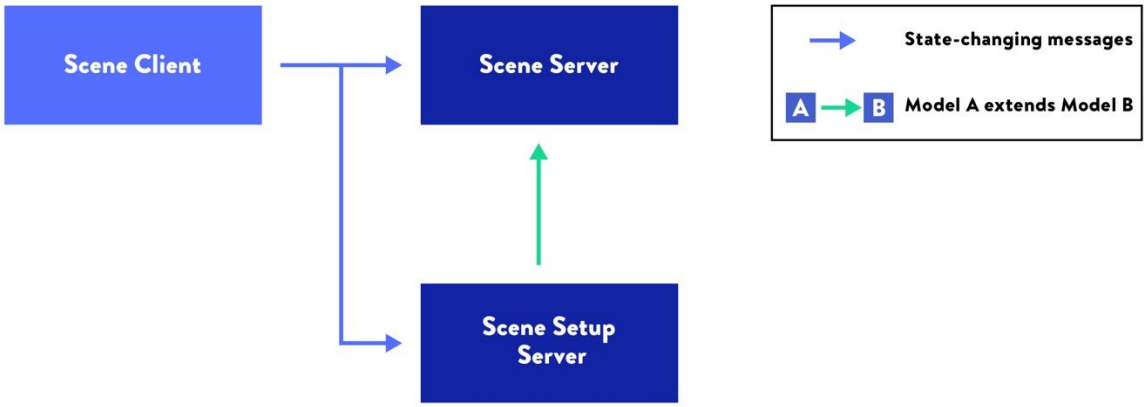

Figure 6.3: Relationships between the models described in this section

7 Sensors

To make the installation aware of what is going on inside the space and to automate certain lighting control processes, we'll add occupancy and ambient light sensors. A connected lighting network that is sensor-driven is capable of tracking changes in the environment and responding to those changes. In a number of applications, this can generate tangible benefits by increasing the overall efficiency of the system. Advanced lighting control strategies based on occupancy and ambient light sensors are also often required by building energy codes. The Mesh Model Specification [1] includes mesh models that support both occupancy sensing and daylight harvesting. The models come with multiple configurable parameters and properties, enabling desired adjustments in response to changing lighting needs or evolving environmental requirements.

Sensors are defined in the Mesh Model Specification [1]. Practically speaking a *sensor* is a device that publishes information. Sensors do not control devices directly. Whether the sensors monitor temperature, humidity, or ambient light levels, they all do basically the same job: they report the current state of certain environmental conditions.

For occupancy sensing, there are different types of sensing devices available on the market. In addition to motion sensors and presence sensors, there are also people-counting sensors that can estimate the number of persons occupying a given space. These three different types of sensors report different values; Bluetooth mesh is designed to support all of them. As far as ambient light sensors are concerned, the Mesh Model Specification [1] allows the sensors to report illuminance measurements from a vast range of 0.00 lux to 167,772.16 lux, and be accurate to the 0.01 lux.

From the perspective of the publish-subscribe paradigm, sensors are not much different from on/off switches or dimmers. The main difference is that switches and dimmers are designed for human use, allowing people to directly control the lighting environment. Sensors, on the other hand, introduce a certain degree of automation to the system. They automatically publish information, and the luminaires respond to that information accordingly. The frequency of such reporting can be adjusted depending on how often you want a particular sensor to check and publish relevant information.

However, in certain cases, periodic reporting is not sufficient to achieve the desired results. That is why the Mesh Model Specification [1] allows implementing a procedure that increases the frequency of reporting when a sensor reading exceeds a certain lower or upper threshold, or when the rate of change of the sensor reading exceeds a certain value.

Sensor models defined in the Mesh Model Specification [1] also have multiple configuration properties.

All of these features provide design flexibility, allowing connected lighting installations to be adjusted to specific requirements of particular premises or to changing requirements of building energy codes. All sensor-relevant settings can be adjusted in the Sensor Setup Server model, which extends the Sensor Server model. Therefore, both occupancy sensors and ambient light sensors need to implement these models. To be able to respond to sensor information, a luminaire needs to implement a lightness controller, which is manifested by the inclusion of the Light LC Server model.

Once a sensor infrastructure is in place, you need to implement certain rules that trigger specific behaviors based on sensor readings. This is what the controller, described in Section 8, is all about.

7.1 Summary of sensor functionality

In this chapter, we described how Bluetooth mesh networking technology enables the use of sensors to provide information about the environment. This information can for instance be used to control the lights within a lighting control system. Table 7.1 below summarizes the configuration for the involved nodes.

Nodes Involved	Luminaires with embedded controllers	Sensors	Configuration Devices
Models Used	Light LC Server	Sensor Server	Sensor Client
	--	Sensor Setup Server	
Publish / Subscribe	Subscribe	Publish	--

Table 7.1: Configuration summary for sensors

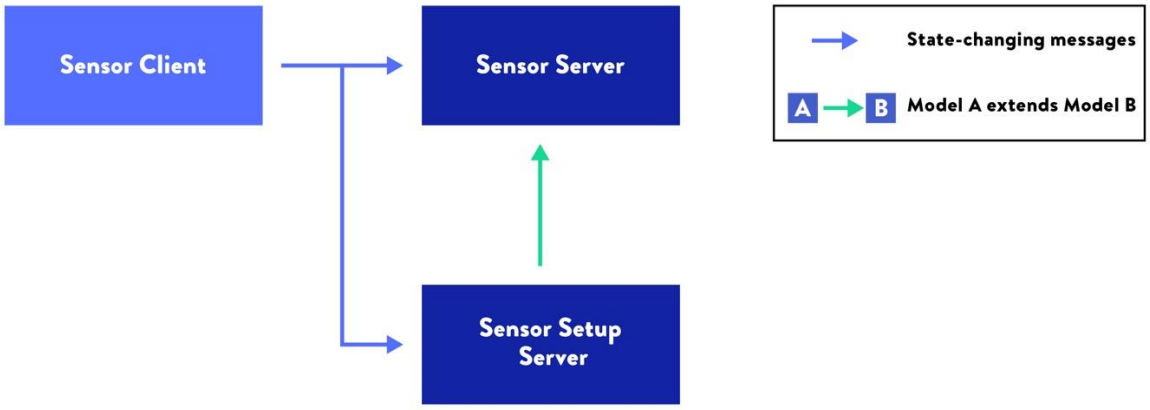

Figure 7.1: Relationships between the models described in this section

For more detailed information on organizing specific functions within multisensors, see Section 10.

8 Controller

The controller is a piece of software that typically sits inside the luminaires that belong to a connected lighting network based on Bluetooth mesh. The Light Lightness Controller controls the luminaire's light level based on information obtained from occupancy and ambient light sensors (ALS). In lighting networks based on Bluetooth mesh, dedicated controller boxes are not needed. The architecture is decentralized and does not have the vulnerability of a single point of failure. There is no need for hardware controllers because software can provide all required capabilities to govern the operation of the entire installation.

Figure 8.1 illustrates the decentralized architecture of Bluetooth mesh lighting control systems. Individual sensors publish information to desired luminaires, or groups of luminaires, without a central hub in between the sensors and luminaires.

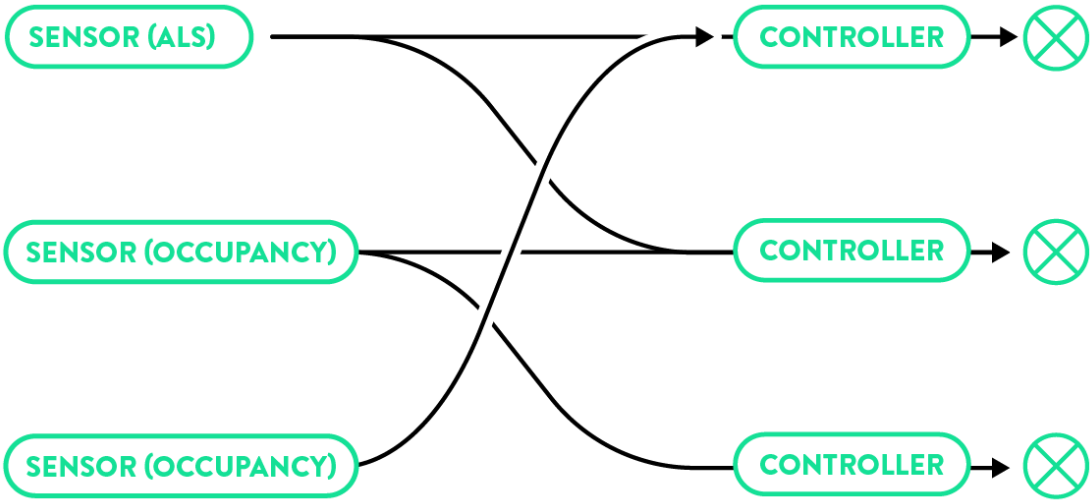

Figure 8.1: Peer-to-peer topology of a Bluetooth mesh lighting network; the sensors publish to groups of luminaires with built-in software controllers

Of course, a device failure can still create disruption. If a malfunctioning occupancy sensor does not provide reliable data, the installation will not work as expected. However, this risk can be reduced by building redundancy into the system, i.e., by using more sensors to control the light level in a location. Multiple sensors can publish information to the same group of luminaires so that if one of the sensors fails, reliable sensor data is still published. Multiple ambient light sensors can also be used this way, although in that case, it is important that all of the sensors provide similar readings. To achieve this, you can install the sensors next to each other.

As noted earlier, the Light Lightness Controller typically resides inside a luminaire. However, because luminaires are multi-component devices, you might wonder where exactly the controller can be found. The precise location may vary depending on the implementation. The configuration of the entire system typically is delegated to a configuration client. The configuration client, in its simplest form, could be just a phone application.

The diagrams in Figure 8.2 and Figure 8.3 show two examples of practical controller implementations. Other arrangements are also possible. For example, controllers can be integrated into multisensors. The controller implementation in Figure 8.2 includes a Bluetooth mesh-enabled LED driver. In Figure 8.3, the implementation incorporates a mesh bridge device. The bridge converts Bluetooth mesh messages to a standard 0 V to 10 V or DALI (Digital Addressable Lighting Interface) output [5], enabling existing lighting

installations to be easily turned into Bluetooth mesh-enabled systems. In both diagrams, the green lines within the luminaire represent local wired connections between individual components. Each individual component is represented by a mesh model. This allows the component, such as a sensor, to be configurable by a Configuration Manager, and to work with both the local luminaire as well as other luminaires subscribed to it. The blue dotted lines represent wireless communication between nodes of a mesh network.

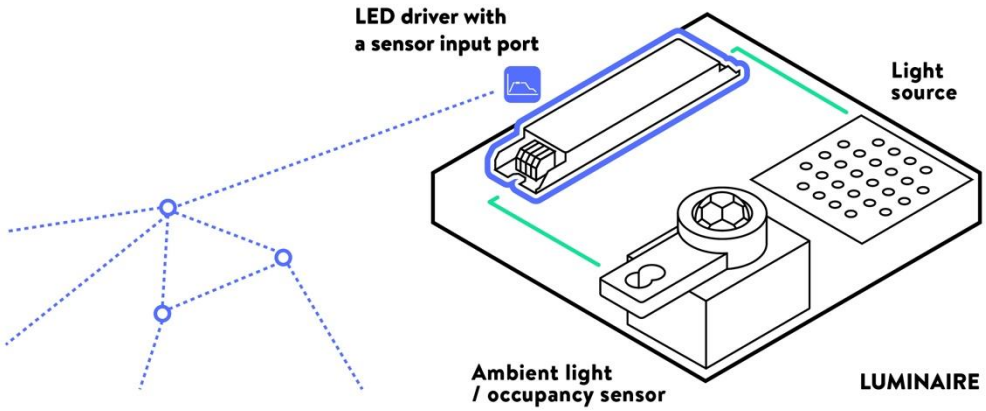

Figure 8.2: Sample controller implementation: a mesh-enabled LED driver with an integrated software controller (marked in blue) and a sensor input port

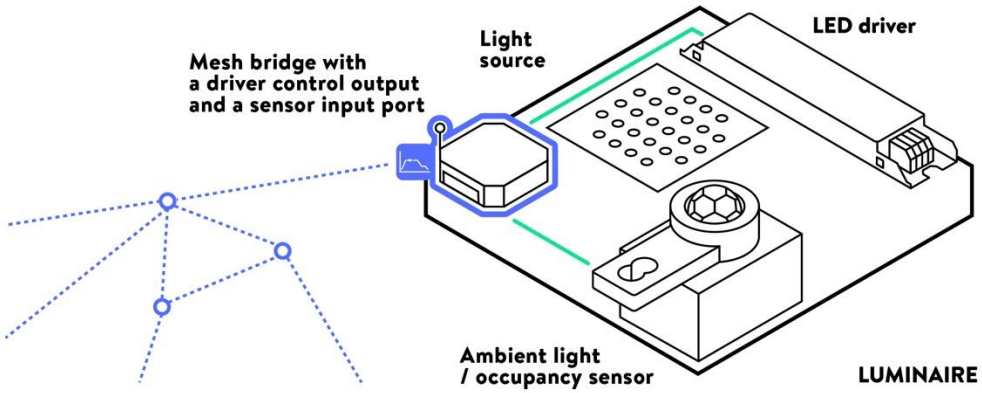

Figure 8.3: Sample controller implementation: a mesh bridge device with a software controller (marked in blue), a sensor input port, and a driver output port

The Light Lightness Controller requires three models: the Light Lightness Control (LC) Server model, the Light Lightness Server model, and the Light LC Setup Server model. Figure 8.4 presents a typical operation of the Light Lightness Controller based on occupancy sensor readings.

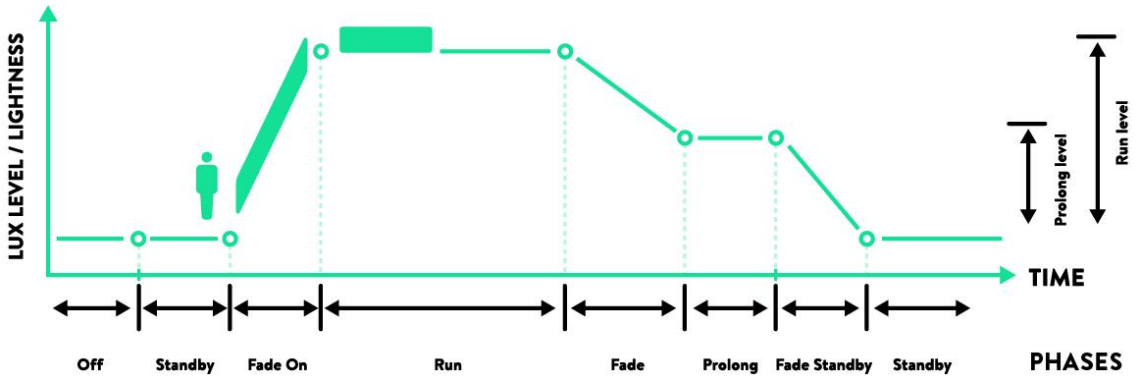

Figure 8.4: Typical operation of a Light Lightness Controller based on occupancy sensor readings

The Light Lightness Controller can be in either the On or Off state. When the Light Lightness Controller is off, it does not control the lighting installation. When turned on, the Light Lightness Controller enters the Standby mode, where it is ready to respond to events from sensors or switches. The events trigger the progression of the Light Lightness Controller through a number of phases, as illustrated by the Figure 8.4. When a sensor detects occupancy in a given area, the Fade On phase is triggered, and the output of luminaires increases gradually to a desired level. When that level is reached, the Run phase begins.

The thick green line in Figure 8.4 represents the period during which the sensors are detecting movement. The light level remains unchanged for a specified period after the sensors last detected movement. After the Run timer expires, the light level gradually decreases (the Fade phase) to the Prolong phase. Again, the Prolong phase lasts for a specified period, after which the Light Lightness Controller starts gradually decreasing the light level (the Fade Standby phase). When this transition ends, if no occupancy was detected in the meantime, the controller returns to the Standby mode.

Each double-headed arrow represents a property that can be adjusted as needed, e.g., to comply with applicable building energy codes. Horizontal arrows represent settings adjusting the duration of relevant phases; vertical arrows represent settings adjusting desired light levels. These levels can be hard-configured by an installer, or they can be stabilized with the help of ambient light sensors. The calculations used in this lightness stabilization process are included in the Light LC PI (Proportional-Integral) Feedback Regulator. The Mesh Model Specification [1] also allows the adaptive control strategies to be overridden by end users whenever needed.

8.1 Summary of controller functionality

In this chapter, we learned how Bluetooth mesh networking technology enables automatic control of the lights within a lighting control system based on sensor inputs such as occupancy or ambient light level sensors. Table 8.1 below summarizes the configuration for the involved nodes.

Nodes Involved	Luminaires with embedded controllers	Sensors	Configuration Devices
Models Used	Light LC Server	Sensor Server	Sensor Client
	--	Sensor Setup Server	
	Light LC Setup Server	--	Light LC Client
	Light Lightness Server	--	Light Lightness Client

	Light Lightness Setup Server	--	
Publish / Subscribe	Subscribe	Publish	--

Table 8.1: Configuration summary for lighting controllers

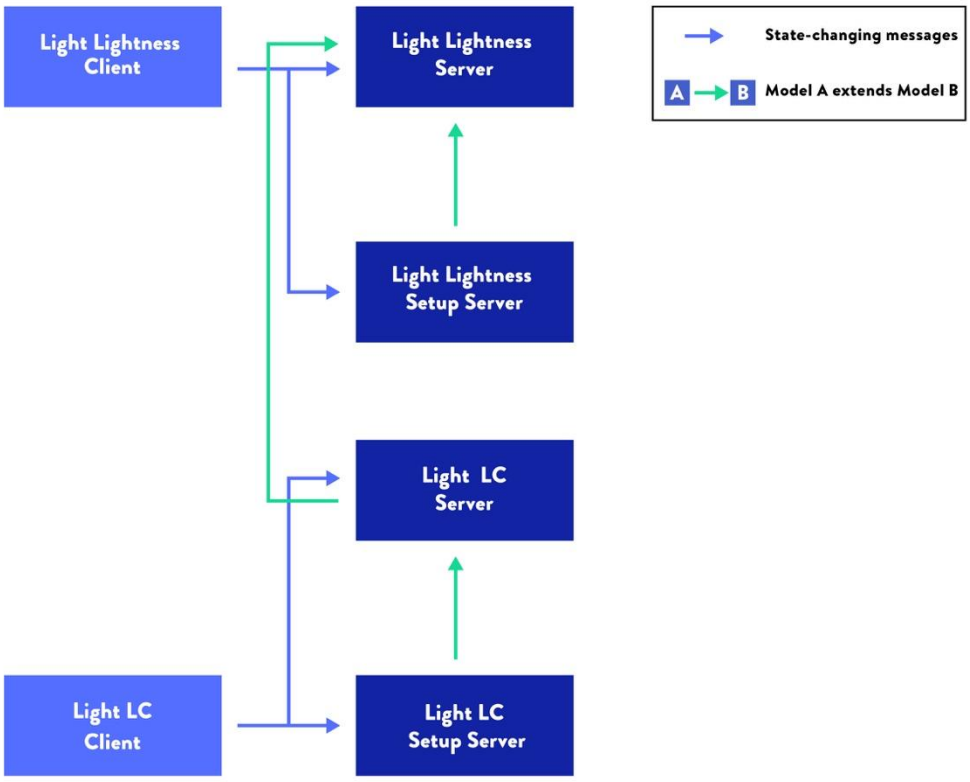

Figure 8.5: Relationships between the models described in this section

9 Lighting control scenarios

The idea of the Light Lightness Controller is to add intelligence to a luminaire. Without the Light Lightness Controller, a luminaire based on the Light Lightness Server is only capable of reacting to messages sent by client models; it cannot control the lightness on its own.

The Light Lightness Controller changes that by allowing a luminaire to understand messages published by sensors (e.g., occupancy sensors or ambient light sensors) and adjust the lightness output accordingly (e.g., to balance the daylight coming through windows and maintain a constant illuminance level when people are present in a room). In common applications, the inclusion of the Light Lightness Controller within a luminaire can eliminate the need for a central “control box,” which is typically present in legacy lighting control systems. With the Light Lightness Controller, the luminaires themselves are intelligent and can form a lighting control system without the need for a centralized controller.

The scenarios supported by the Light Lightness Controller can be implemented to help increase energy efficiencies and to customize lighting systems to meet the requirements of building energy codes. Implementations also can be fine-tuned to address specific lighting needs. Before diving into the lighting control scenarios, let’s first consider what is a luminaire with an integrated controller.

A luminaire with an integrated controller is a two-element Bluetooth mesh node. The controller’s Light LC Server controls the lightness of the luminaire that is implementing the Light Lightness Server model through a binding between the Light LC Linear Output state and the Light Lightness Linear state. Each of the two elements can receive messages independently, so the luminaire either can be controlled by the controller or it can directly receive messages from other devices. This arrangement is illustrated by Figure 9.1 where the light level of the luminaire is controlled by connecting the Light LC Linear Output state of the controller with the Light Lightness Linear state of the luminaire. For a discussion of the configuration of elements and models in devices on a mesh network, see the Mesh Profile Specification [2], Section 2.

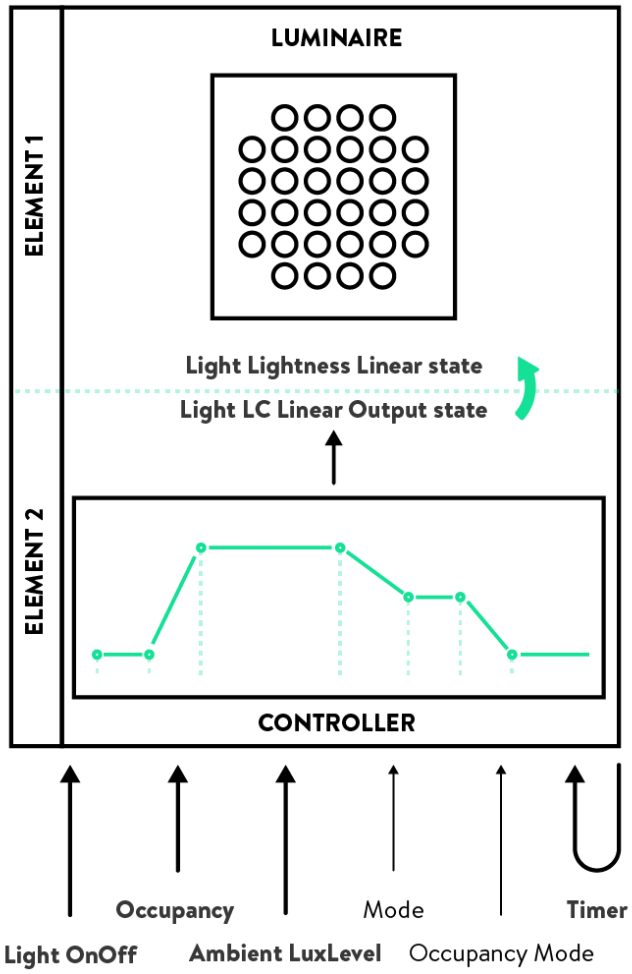

Figure 9.1: Overview of a luminaire with an integrated controller

By principle of its operation, the output of the proportional-integral regulator is linear, which is why the Light Lightness Linear state is used for binding.

The controller has a total of six inputs. These include three control inputs (Light OnOff, Occupancy, Ambient LuxLevel); two mode inputs (Mode, Occupancy Mode); and the internally generated Timer. The inputs allow for choosing specific events that trigger the controller to transition between its individual phases. These events relate to switch operation, occupancy detection, or timer expiration. The output from the Light LC controller is the Light LC Linear Output state, which is conditionally bound to the luminaire's Light Lightness Linear state. The Mode input is represented by the Light LC Mode state. The Light LC Mode state is a binary state that determines whether the controller is on or off. The Light LC Mode state can be controlled by Light LC Mode messages. When the controller is off, the binding between the controller's Light LC Linear Output state and the luminaire's Light Lightness Linear state is disabled, and the controller does not control the lightness of the luminaire. Instead, the lightness can be controlled by the Light Lightness Client model that is implemented, e.g., by a wall switch. When the controller is on, the binding is enabled, and the luminaire's output is driven by the controller.

The controller's Light LC Mode state is bound to an instance of the Light Lightness Linear state of a luminaire. This binding monitors whether any of the Light Lightness states are changed directly (i.e., not by the controller itself). The concept is similar to a cruise control feature in a vehicle. A cruise control system turns off automatically when a driver depresses a brake. Similarly, a luminaire cruises in accordance with the controller configuration as long as the controller remains on. But when the luminaire

is turned off or is dimmed directly—using, for example, a wall switch—then the controller's Light LC Mode state is automatically set to off, and the controller no longer controls that luminaire. To put the luminaire in cruise mode again, the binding between the controller's Light LC Mode state and an instance of the Light Lightness Linear state needs to be restored (via an explicit message). This arrangement allows for overriding any sensor-driven lighting control scenarios by using, for example, wall switches, as illustrated in Figure 9.2. There, the green arrow from the switch to the luminaire illustrates the direct control of the luminaire by the switch. In addition, the red dotted arrow pointing towards the connection between Light LC Linear Output state of the controller and Light Lightness Linear state of the luminaire means that this connection will no longer be active once the switch is used, i.e. the controller no longer controls the luminaire in that case.

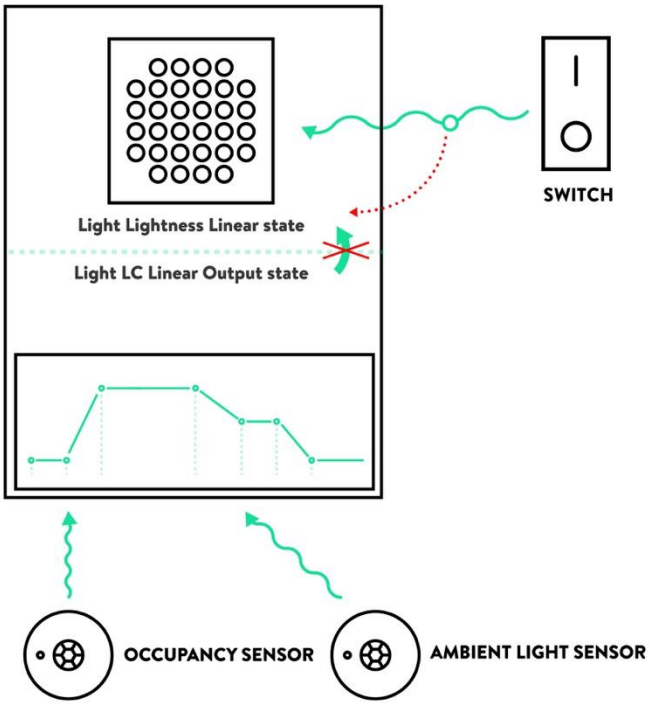

Figure 9.2: Overriding the current lighting control scenario with a wall switch

When the controller is on, the luminaire integrated with the controller transitions through successive phases of the controller's operation as illustrated by Figure 8.4. These transitions can be triggered by several groups of events, including the following event types:

Light On/Off events, occurring when on/off messages are received, for example, from switches

Occupancy events, occurring when occupancy sensors detect human presence in a given area

Timer events, occurring when the value of the countdown timer reaches zero

The Light Lightness Controller enables implementation of three different lighting control scenarios. These are: switch control, vacancy sensing, and occupancy sensing. Even though they are based on a very similar procedure, they achieve very different results. Each of them also uses the controller's state diagram illustrated by Figure 8.4.

9.1 Switch control

The switch control scenario does not involve any sensors, although it does use the controller to implement a certain degree of automation. In this scenario, switches are used to turn the luminaires on for a specified period of time. After this time expires, the luminaires turn off, although a switch can be used again in the meantime to restart the timer. In practice, this scenario is commonly implemented in stairwells, when no sensors are involved.

To implement the switch control scenario, a relationship between the switch and the controller needs to be established. A luminaire with an integrated controller has two separate On/Off inputs (one on each element). The first On/Off input allows for controlling the luminaire directly, bypassing the controller. The second input controls the Light LC Light OnOff state (bound to the second instance of the Generic OnOff state). This is the input that a switch needs to use to enable the switch control scenario.

To establish such a relationship, the switch needs to implement the Light LC Client model, which publishes to the address to which the controller's Light LC Server model is subscribed. Alternatively, the switch implementing a Generic OnOff Client model may publish to the address to which the controller's Generic OnOff Server model is subscribed. In each case, pressing the ON button on the switch publishes a message that triggers the controller, i.e., makes it enter the Fade On phase. This turns relevant luminaires on and triggers the Run timer. When the Run timer expires, luminaires fade to the Prolong phase and then to off when the Prolong timer expires.

Pressing the switch again before the Run timer expires restarts the timer, extending the Run phase. Pressing the switch during the Prolong phase restarts the Run phase. Figure 9.3 illustrates how the events in this scenario map to the modes in the Light Lightness Controller state diagram illustrated by Figure 8.4.

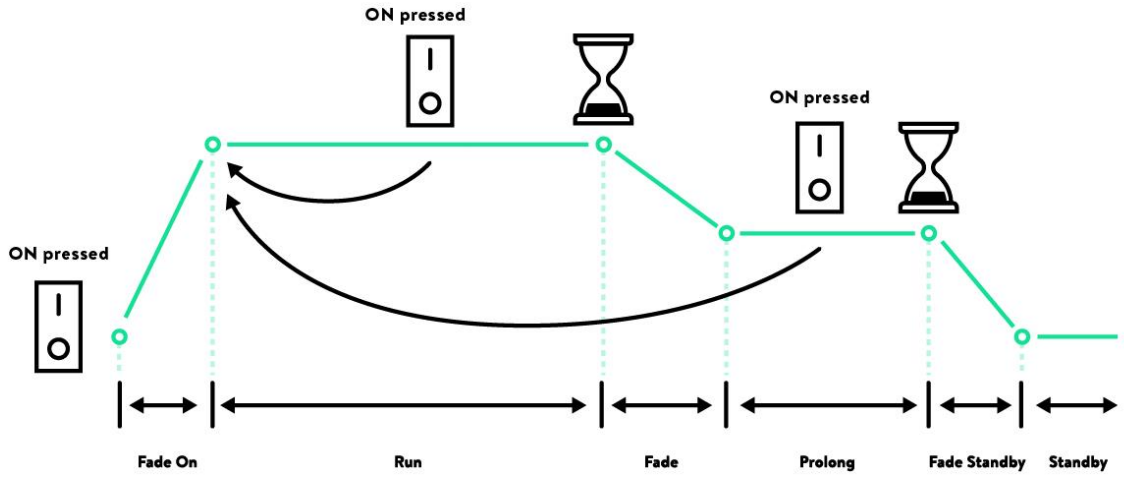

Figure 9.3: The switch control scenario mapped to the controller's state diagram

As was the case with the switches discussed in Section 3.3 (momentary switches), the switch in the switch control scenario also does not have a state. It is a momentary switch, and pressing it triggers the publication of a message. That message sets the Light LC Light OnOff state of the controller. Figure 9.4

illustrates the on/off switch operation when controlled by the controller (as compared with the simple on/off switch operation in Figure 3.6).

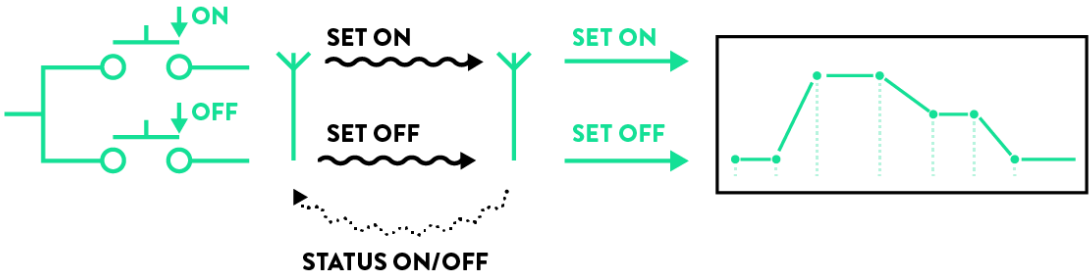

Figure 9.4: Triggering the Light Lightness Controller with a wall switch

Contrary to messages that directly set the Generic OnOff state of the luminaire, messages setting the Light LC Light OnOff state of the controller do not interrupt the controller's operation. They just trigger the controller's transition to the Run phase or to the Standby phase. The controller remains engaged as long as messages are delivered through its Light OnOff input.

9.2 Vacancy sensing

This scenario introduces occupancy sensors to turn off the light when the area is vacant, although switches are still used to trigger the controller. Similar to the switch control scenario, pressing a switch turns luminaires on for a predefined period of time. However, in this scenario, the controller is also processing data from occupancy sensors. Each occupancy detection restarts the Run timer, so the light stays at the configured level as long as occupancy has been detected within the most recent period defined by the timer.

When the Run timer that was triggered by the most recent movement detection expires, the luminaires fade to the Prolong phase. When occupancy is detected while in the Prolong phase, the controller reenters the Run phase (entering the Fade On phase first, but starting at the light level of the Prolong phase) and restarts the Run timer as indicated by the arrow pointing from the Prolong phase to the Run phase in Figure 9.5 below. After the Prolong timer expires, the controller enters the Standby phase, and a switch is used to trigger the controller—and the luminaires—again. Figure 9.5 illustrates how the vacancy sensing scenario maps to the controller's state diagram illustrated by Figure 8.4.

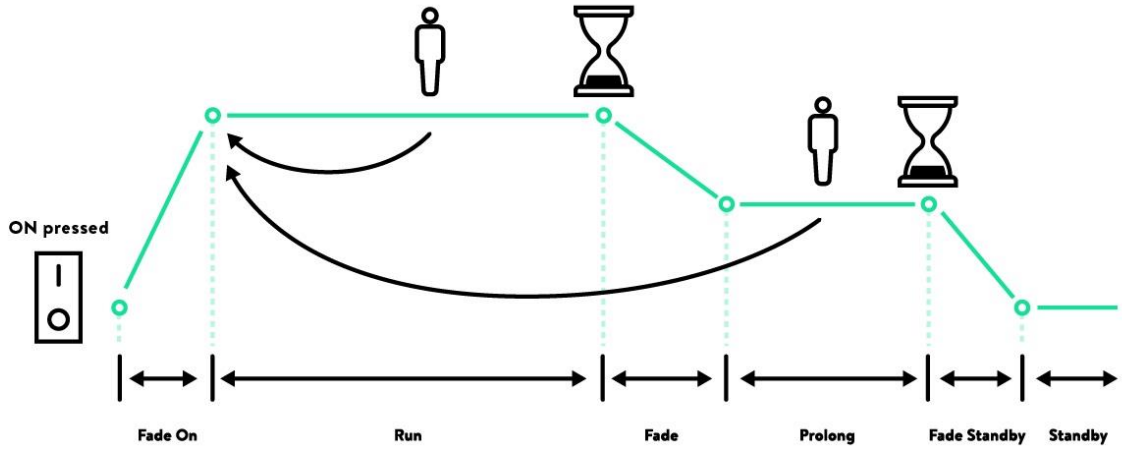

Figure 9.5: The vacancy sensing scenario on the controller's state diagram

Similarly, as in the switch control scenario, a relationship between a switch and the controller needs to be established. In addition, the controller's configuration has to allow for processing sensor data received through the Occupancy input. The input is represented by the Light LC Occupancy state and accepts data from one or more sensors reporting the Occupancy Property via Sensor Status messages.

The controller's Light LC Server model is used to represent an element that is a client to a Sensor Server model implemented by an occupancy sensor. So, to be able to process the sensor data, the controller (the Light LC Server model on a luminaire) needs to be subscribed to the address to which the sensor (the Sensor Server model on the sensor device) publishes. Finally, to enable the vacancy sensing scenario, the Light LC Occupancy Mode state, which represents the Occupancy Mode input, must be set to zero. This way, the controller won't transition from a standby state when occupancy is reported; only the switch can trigger this transition.

9.3 Occupancy sensing

Occupancy sensing is the scenario that provides the highest degree of automation. In its implementation, this scenario is very similar to vacancy sensing. The only difference is that occupancy sensors both trigger the controller and keep it running based on occupancy detection. This scenario may optionally involve switch operation.

The outcome is that luminaires turn on automatically whenever sensors detect occupancy. This also triggers the Run timer, which restarts whenever sensors report occupancy again. Once the timer expires and no occupancy is detected, luminaires fade to the Prolong phase and then to the Standby phase. Figure 9.6 illustrates how the occupancy sensing scenario maps to the controller's state diagram illustrated by Figure 8.4.

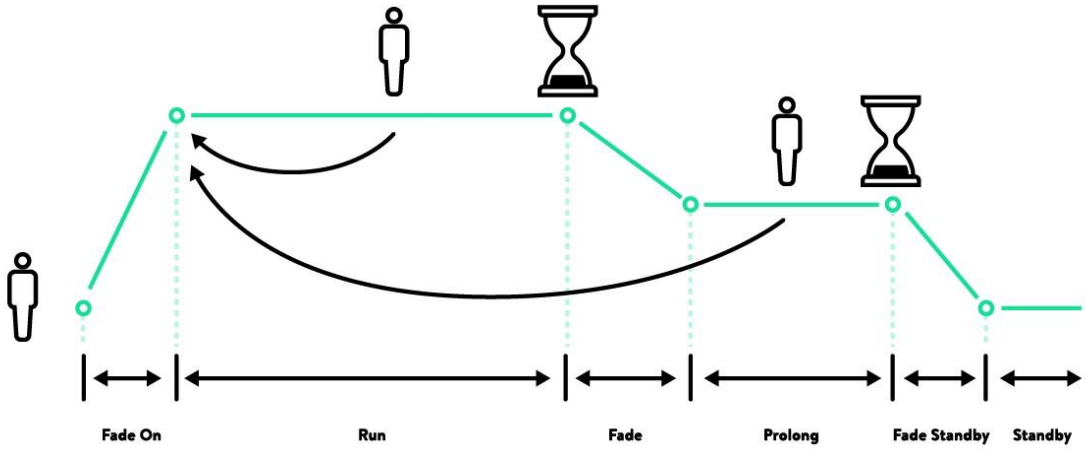

Figure 9.6: The occupancy sensing scenario on the controller's state diagram

Because switch operation is optional in this scenario, there is no need to set up a relationship between the controller and the switch. Sensor data is provided through the Occupancy input, so the configuration of the controller is almost the same as in the vacancy sensing scenario. The only difference is that the Light LC Occupancy Mode state must be set to 1. This allows the controller to transition from the Standby state whenever sensors report occupancy.

9.4 Support for daylight harvesting

The switch control, vacancy sensing, and occupancy sensing scenarios all can support the daylight harvesting feature. Daylight harvesting requires deployment of ambient light sensors that provide the

controller with illuminance data through the Ambient LuxLevel input. This input is associated with a proportional-integral feedback regulator, which stabilizes the light to a configured level.

To enable the daylight harvesting feature, the controller's setpoints must be configured in lux (as opposed to relative levels, when no ambient light sensors are used). The controller then processes data from ambient light sensors to dim the luminaires down or turn them off when lighting needs are, at least partially, met by the natural light entering the room. However, a minimum output of the luminaires still can be specified.

While the proportional-integral feedback regulator adjusts the light to a desired lux level, the minimum output can be specified in relative values. Therefore, it is possible to configure the controller in such a way that an illumination level of 300 lux is always maintained but at the same time the luminaires never dim below 10 percent. Daylight harvesting does not change anything in the way these scenarios are implemented. It is a feature that sits on top of lighting control scenarios.

10 Elements and multisensors: the advantages of multi-element multisensors

Elements are defined in the Mesh Profile Specification [2] states. Mesh network nodes can have more than one element. However, there are many examples of mesh network nodes that only have one element and therefore wouldn't even require the concept of elements to function properly. However, complex devices such as multi-lamp luminaires need to implement elements to reach their full potential in a mesh lighting network. In addition to the *primary element*, such nodes have one or more *secondary elements*. A single Bluetooth radio chip handles all elements of the luminaire.

The most straightforward approach to designing a multisensor would be designing a single-element device. Even though such devices contain a number of different sensors, they transmit all the readings in a single message. The Sensor Server model supports multiple sensors on a single element to avoid extra energy use, which is particularly important for battery-powered sensors. Energy savings are possible because a simple Sensor Status message can carry multiple sensor readings (e.g., ambient light level and occupancy status). However, for more sophisticated lighting control strategies, this may not be flexible enough. Consider the scenario illustrated in Figure 10.1.

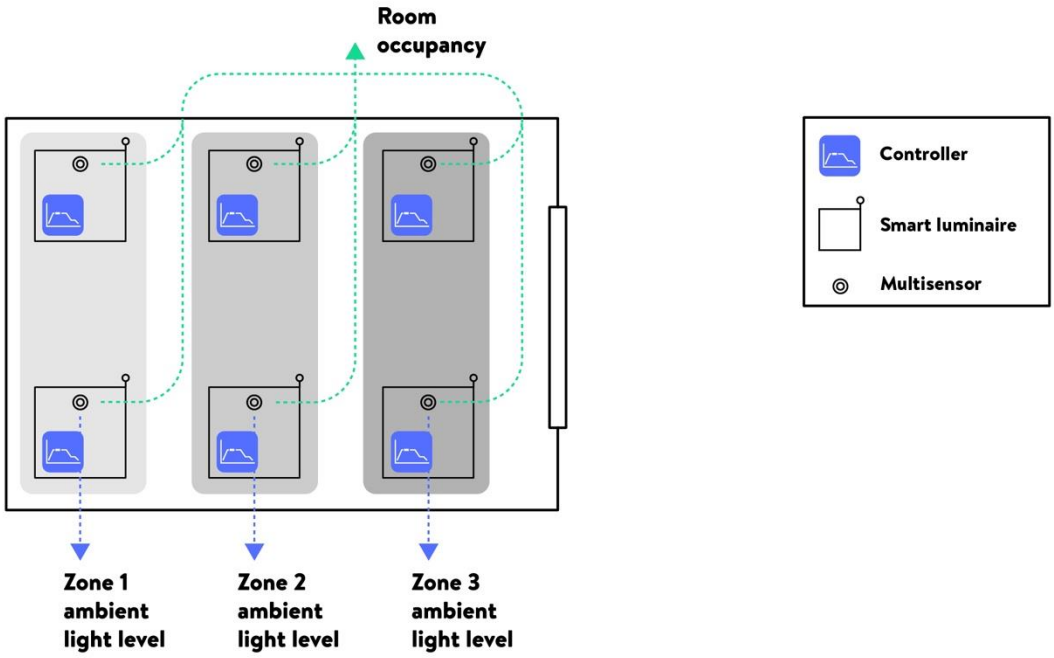

Figure 10.1: Sample lighting control system that features daylight harvesting and occupancy sensing strategies

This small office space has six luminaires, each equipped with a multisensor. Every sensor in the room reports occupancy status so that whenever any of them detects occupancy, the lights across the entire space turn on. At the same time, three different zones have been specified to provide an efficient daylight harvesting implementation. Based on ambient light level readings from individual sensors, the luminaires' output is automatically adjusted for each of the zones to deliver an optimal mix of daylight and artificial light in a given area.

For this particular setup, four group addresses need to be specified: one for all of the luminaires in the room (to respond to occupancy readings) and three for daylight harvesting zones (to respond to ambient

light level readings). A message from a single-element multisensor typically looks like the one in Figure 10.2.

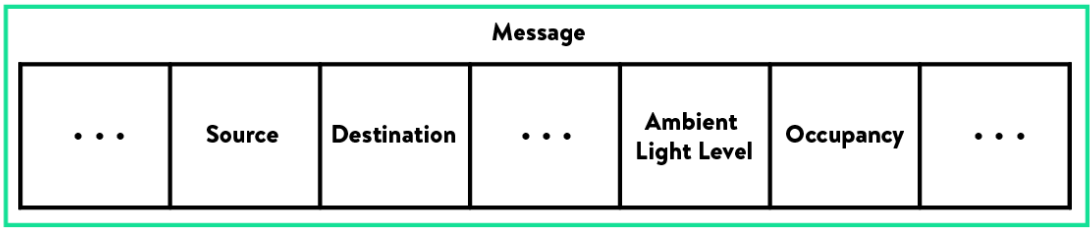

Figure 10.2: A multisensor data message that includes ambient light level and occupancy data

As shown in Figure 10.2, a single message carries the readings from both the ambient light sensor and the occupancy sensor, as well as the source and destination addresses. However, in this example, we need different destinations for occupancy data (“Room”) and for ambient light level data (“Zone 1/2/3”). It is not possible to set the sensor’s Publish Address in such a way that it supports this arrangement. Occupancy readings and ambient light level readings need to be sent to two different addresses, and the only way to do that is to split the message into two separate messages, each with a distinct destination address, as shown in Figure 10.3.

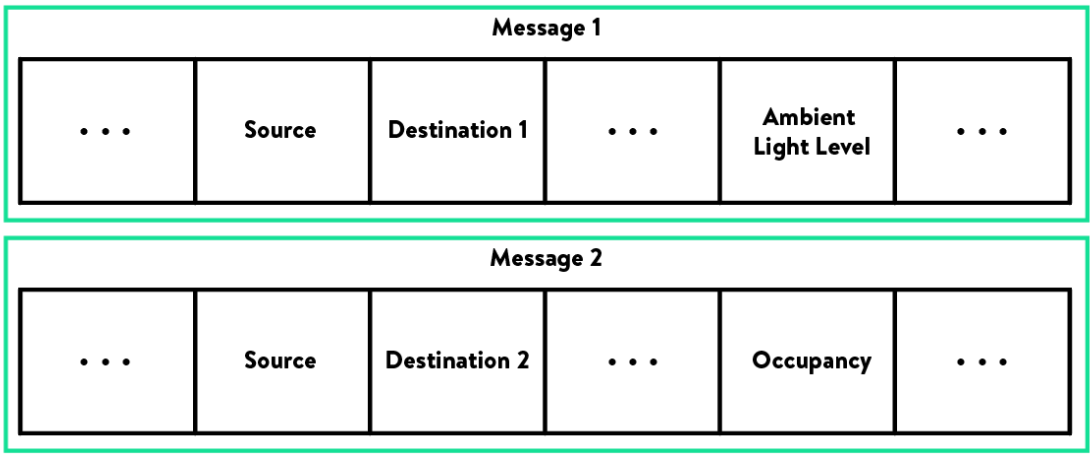

Figure 10.3: Separate data messages for occupancy readings and ambient light level readings

To fully support the capabilities of mesh lighting networks, multisensors used for lighting should be implemented using the multi-element concept to allow publishing different types of sensor readings as separate messages. Figure 10.4 illustrates the difference in how sensor data is handled by a single-element multisensor and a multisensor implementing the multi-element concept outlined in the Mesh Model Specification [1].

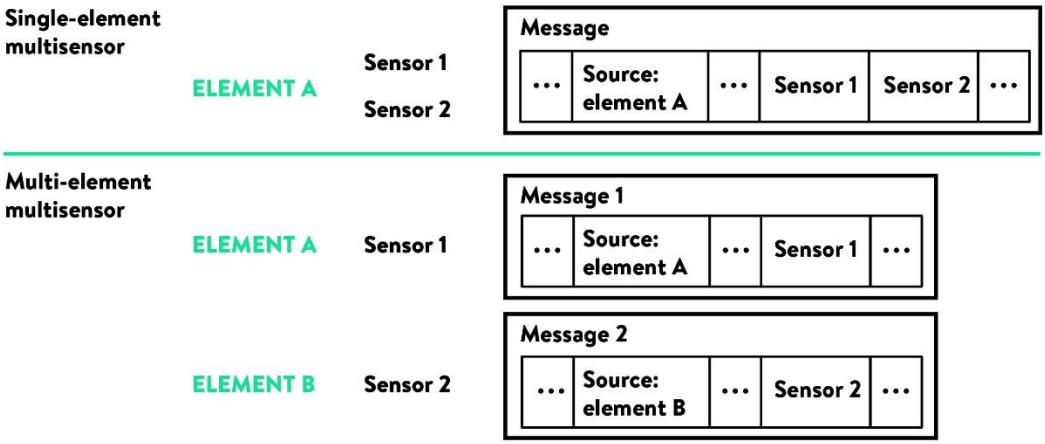

Figure 10.4: Sensor data being handled by a single-element multisensor and a multisensor implementing the multi-element concept

Loc field: When discussing the concept of elements, we should briefly mention the Loc field, which is one of the element description fields contained in the Composition Data state of a mesh device. The Composition Data state contains information about a node, the elements it includes, and the models it supports. The Loc field, in particular, contains a location descriptor. The location can be specified using one of many values defined in the GATT Namespace Descriptors section of Bluetooth SIG Assigned Numbers [4]. The section includes such descriptors as “auxiliary”, “main”, “bottom”, “upper”, “external”, “front”, “back”, “first”, “second”, and “third”.

A properly filled Loc field provides users with more precise information that facilitates configuration activities. A phone app can, for example, display the following message: “You are now configuring the first button of the switch.” The Loc field also allows for separate classification of upper and lower lights or components, which can help to avoid confusion during their configuration. The content of the Loc field could also suggest specific functionalities of a particular device, to help improve the overall configuration experience for customers. Therefore, to facilitate configuration activities, it is beneficial for the Loc field to contain a clear and precise element description.

Appearance characteristic: Further information can be included in the Appearance characteristic, which represents the external appearance of the device. The Appearance characteristic is introduced to make it easier to match a device found during device discovery with the physical device. Appearance information can be provided by an unprovisioned device, by adding an «Appearance» AD Type (described in the Mesh Profile Specification [2]). This is particularly useful during the provisioning process, as it allows for quick identification of the device in the commissioning app. The device manufacturer decides on which appearance to use (out of the set defined by the Bluetooth SIG).

11 Building a complete installation

At this point, we have all the components that we need to build an adaptive, sensor-driven, connected lighting system. To set up such an installation, we'll need to make sure that each network node implements the required mesh models as described in all the previous sections. This means that individual groups of devices need to implement the models listed in Figure 11.1.

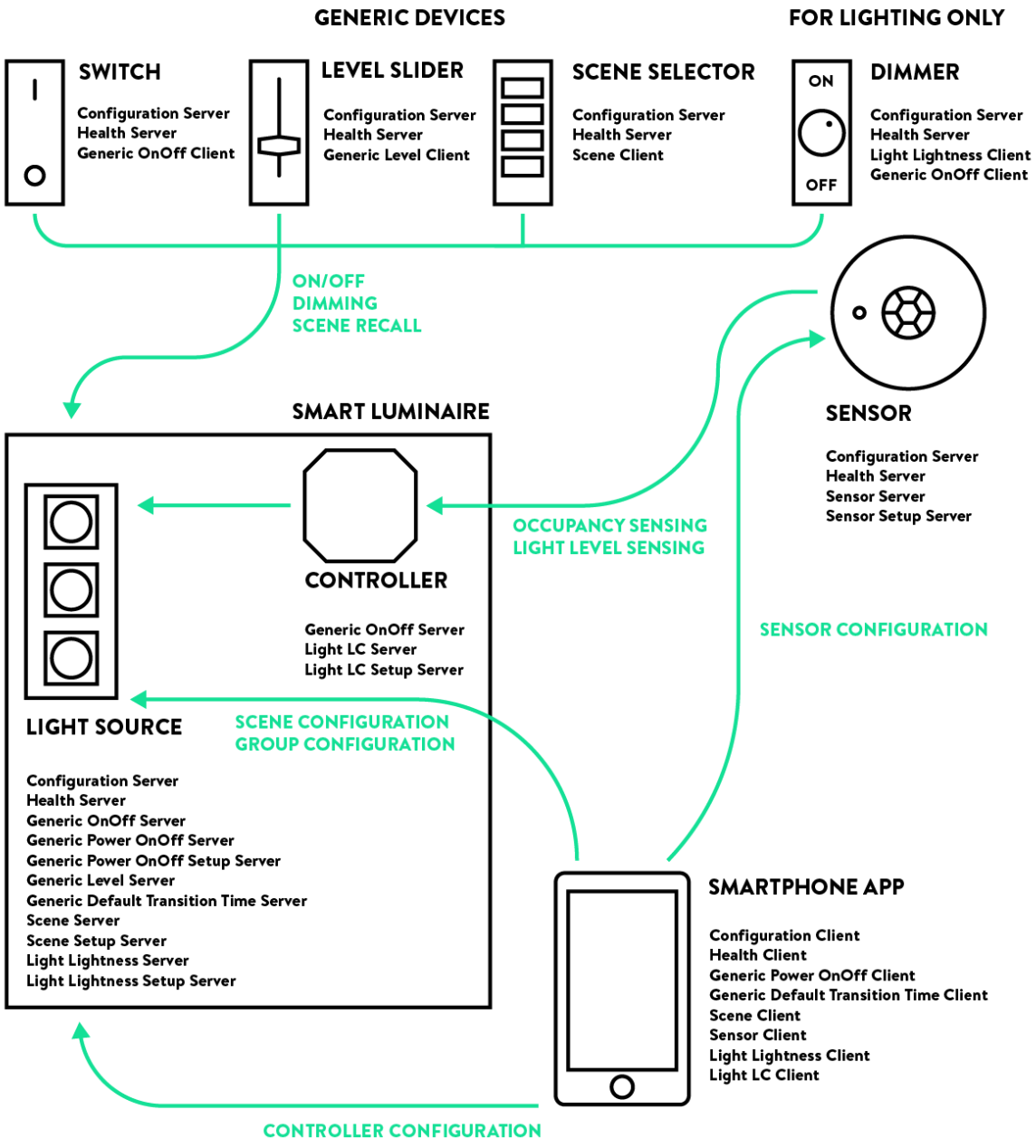

Figure 11.1: A complete list of models required to set up a lighting control system with occupancy sensing and daylight harvesting features

To operate such a network in accordance with specified requirements (e.g., applicable building energy codes), relevant properties—such as transition times, cadences of sensor reports, duration of controller phases, or required light levels—can be adjusted to the desired values.

In practice, a Configuration Manager is also required to set up a mesh lighting network. The Configuration Manager is a device that can add (i.e., provision) new devices onto the network, also providing them with the necessary application keys and assigning publish and subscribe addresses.

An unprovisioned device cannot send or receive mesh messages. However, the device can advertise its presence to a Configuration Manager. After a new device is authenticated, a Configuration Manager can invite the device into a mesh network so that it becomes a network node. After application keys are delivered and publish/subscribe addresses are assigned to the node, the device can communicate with other nodes of a mesh lighting network.

Because the Bluetooth radio can be found in virtually any smartphone or tablet on the market, such devices can be conveniently used as Configuration Managers—as long as they have a relevant app installed that supports all the required processes. A smartphone app can also be used to configure groups and scenes, and to adjust any operational settings. In addition, a cloud platform may be needed to store the network configuration backup, and to allow other authorized persons/devices to reconfigure the network or adjust its parameters. Developing efficient, intuitive, and user-friendly commissioning and maintenance tools is a challenging task, because the design of these tools has a profound impact on the overall connected lighting experience.

The scope of functionalities described in this white paper represents only a few of the possibilities enabled by mesh models included in the Mesh Model Specification [1]. The specification also addresses issues such as time synchronization, allowing for an implementation of precise time scheduling. As the system that you want to build becomes more sophisticated, the number of mesh models that the system uses increases—but the instructions and paradigms described here should be a good starting point for building complex mesh installations.

12 References

[1] Bluetooth Mesh Model Specification, Version 1.0 or later

[2] Bluetooth Mesh Profile Specification, Version 1.0 or later

[3] Bluetooth Core Specification, Version 5.0 or later

[4] Bluetooth SIG Assigned Numbers, http://www.bluetooth.com/specifications/assigned-numbers

[5] IEC 62386 - the international standard for DALI technology, https://www.digitalilluminationinterface.org/dali/standards.html