SILVAIR

How to build a wireless sensor-driven lighting control system based on Bluetooth mesh networking

White paper
N° 1

Revision **1.0**
September 2017

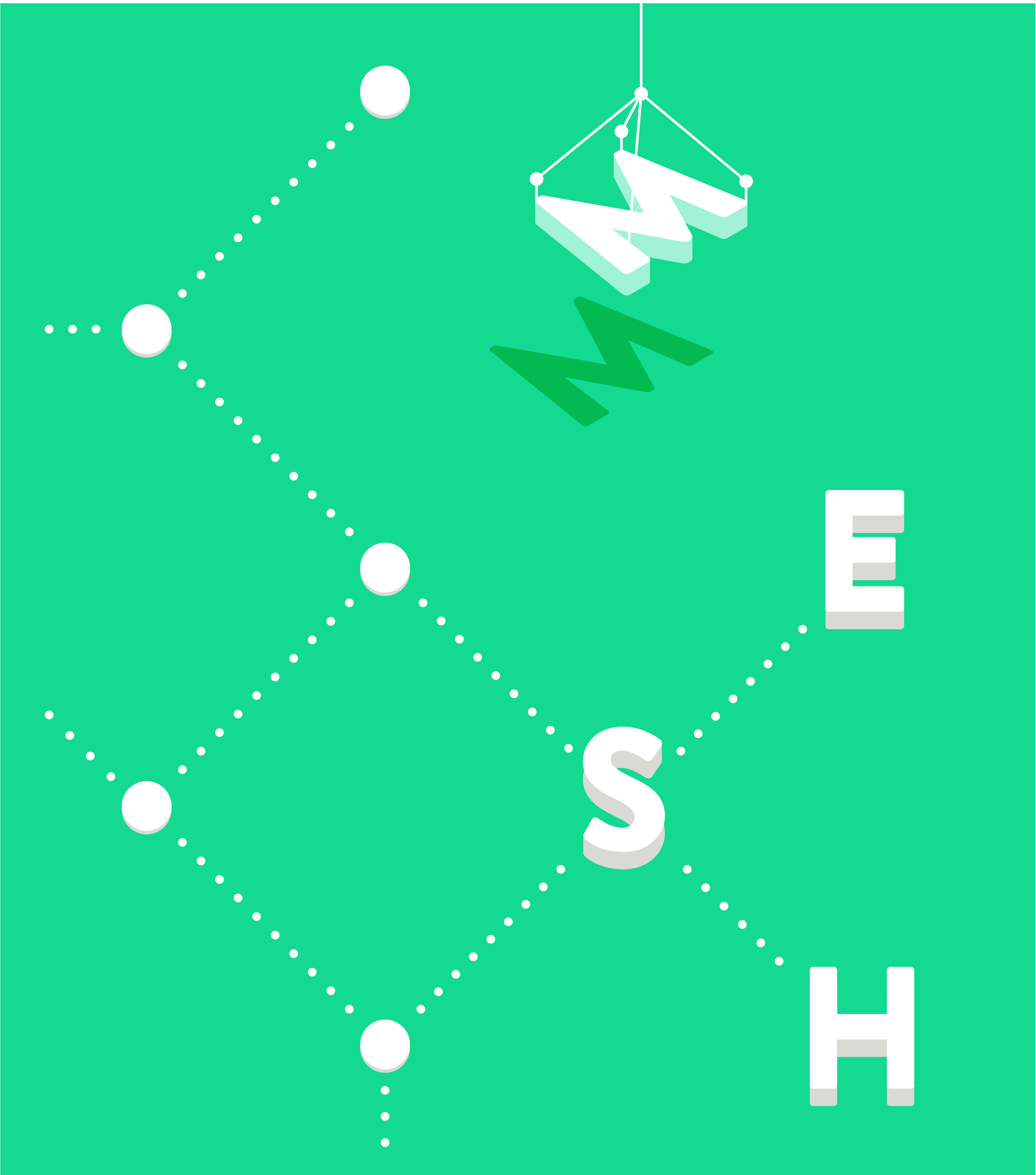


Table of Contents

Abstract 3

Introduction 4

On/off switch 4

Light level control 6

Transitions 8

Scenes 9

Sensors 10

Controller 11

Elements and multisensors 13

Building a complete installation 13

Abstract

Bluetooth mesh networking technology allows for building flexible smart lighting networks that use sensors to drive efficiencies. Its specification outlines a number of mesh models defining basic functionalities of network nodes. This paper explains which mesh models need to be used in different types of lighting and sensor devices, while also describing how these devices can be set up to operate in concert and form adaptive, highly efficient connected lighting networks.

Introduction

By defining basic functionalities of network nodes, mesh models address the problem of interoperability at the application level. They make devices aware of what function they perform, what other nodes they can connect with, and what actions can be performed upon them. To enable maximum design flexibility, models include multiple properties that can be adjusted in accordance with specific applications. Simple devices can rely on very few models, but more complex ones need to employ many of them. This results from the fact that certain models require others that must be instantiated within the same node. This is called model extending.

Each device must implement two Foundation Models in order to function properly within a wireless mesh network. These are the Configuration Model (used for network management) and the Health Model (used to represent mesh network diagnostics of a device). The Foundation Models are defined in the **Mesh Profile Specification**, while all the other models referred to in this paper are defined in the **Mesh Model Specification.**

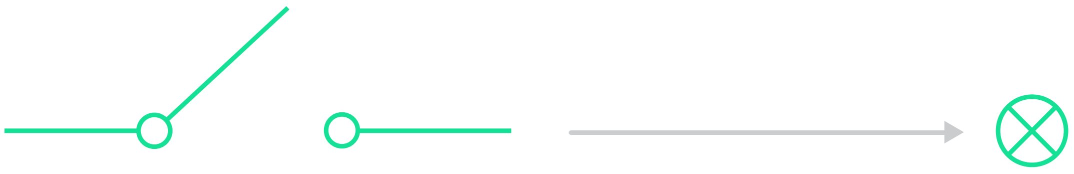
To illustrate how a complete adaptive lighting system can be set up using the Bluetooth mesh networking technology, we'll first illustrate how to build individual devices with mesh connectivity - starting with the most basic concepts and progressing to more sophisticated elements. Mesh models will be the building blocks enabling us to incorporate more and more complex functionalities into individual network nodes. Once we have all the necessary components in place, we'll be able to build a complex, sensor-driven system.

On/off switch

Since a lighting control system is what we are going to build, a light source is our most basic network component - we'll need it in each of the scenarios described below. In traditional lighting systems, a light source is typically controlled with a wall switch. While wireless networks enable more ways to control lights, switches remain primary lighting control tools. They are wireless and more flexible in terms of operational characteristics, yet they can be used exactly the same way as traditional wired switches.

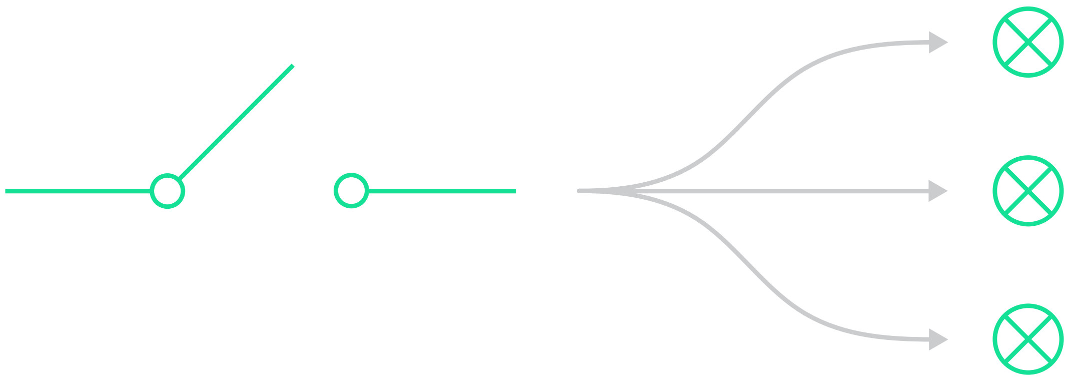
The most basic lighting control scenario involves a light source and an on/off switch. To enable such a simple system, the light source must implement the Generic OnOff Server model, while the switch must employ the Generic OnOff Client model. These are generic models, which means they are nonspecific in their functionality. They can be used in a variety of applications, including lighting.

Before a switch (implementing a client model) can control the light source (implementing a server model), the Publish Address of the switch needs to be set to the lamp's unicast address. The light source itself does not require any specific information. Once the switch knows the lamp's address, it can be used to turn the lamp on/off. This is basically all that is needed to build an extremely straightforward lighting control system involving one on/off switch and one light source. This simple scenario can be shown using the following diagram:



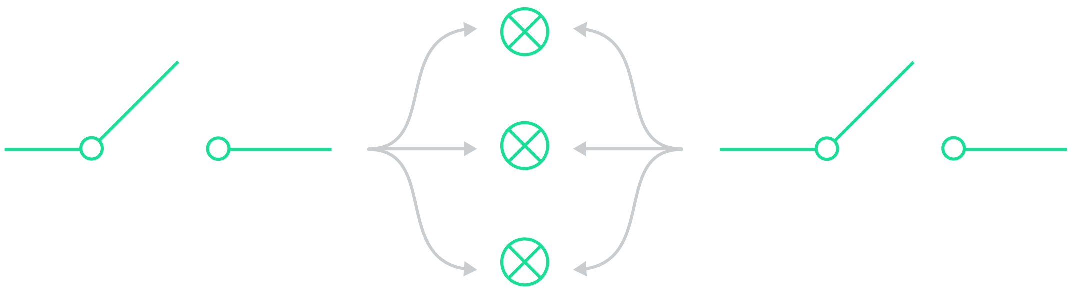
‹ **Figure 1.**
One on/off switch and one light source

But of course switches are rarely used to control single lamps. More often than not, you'll want to control a group of lamps using a single switch. In this case, a group address needs to be created and set as the model's Publish Address. Like in the previous example, this information must be specified for the Generic OnOff Client model used by the switch. In addition, the light sources that are to be controlled this way must act upon messages sent to the entire group. To achieve this, the Generic OnOff Server model implemented by each of them must subscribe to the group address - the same one that has been specified as the Publish Address for the switch. This is how we've set up a group of lamps and an on/off switch controlling all of them simultaneously. A corresponding diagram looks as follows:



‹ **Figure 2.**
One switch and multiple lamps

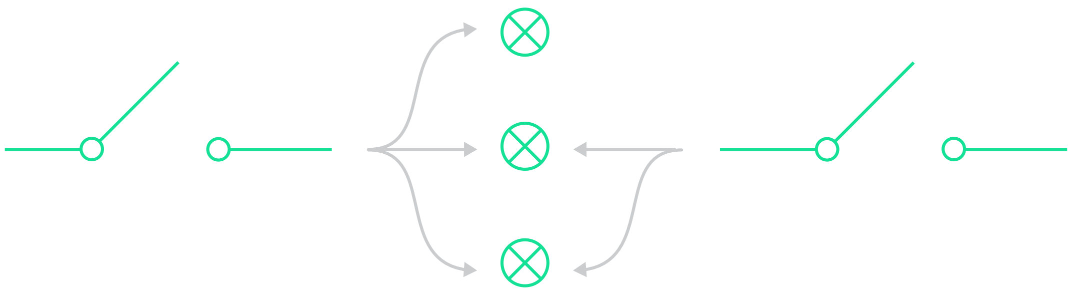
Increasing the number of switches in such installation couldn't be easier. Each additional switch will, again, need to implement the Generic OnOff Client model with the exact same group address that has been specified for other switches (as their Publish Address) and corresponding light sources (as the group address they subscribe to). This way, each additional switch can control the same group of lights independently, as shown in the diagram below:



‹ **Figure 3.**
Multiple switches and multiple lamps

SILVAIR

Finally, we might want to have multiple switches controlling multiple light sources that are assigned to multiple overlapping groups. The following diagram illustrates a simple example of such system:


‹ **Figure 4. Overlapping groups of lamps controlled by multiple switches**

This can be built using the same methods and requirements that we followed when building the previous systems. No additional mesh models are required, either. In this case, there are three light sources and two switches. One of them controls all of the lamps, while another controls only two of them - so there are two overlapping groups. This means we need two different group addresses. The uppermost lamp subscribes to one of them, while the remaining ones subscribe to both groups.

So far, we've built a number of simple networks with on/off switches controlling a number of light sources. We've also learned how to group nodes within a mesh network. The principles we used for grouping can be applied to a number of other applications provided for in the Mesh Model Specification. This means that the majority of scenarios presented in the remaining part of this paper can be implemented in abovementioned configurations. Using these simple examples, we've also learned the basics of the client-server architecture communicating with a publish-subscribe paradigm that is one of the core concepts behind Bluetooth mesh networking.

Quick recap

nodes involved	light sources	on/off switches
models used	Generic OnOff Server	Generic OnOff Client
paradigm	subscribe	publish

Light level control (including dimming)

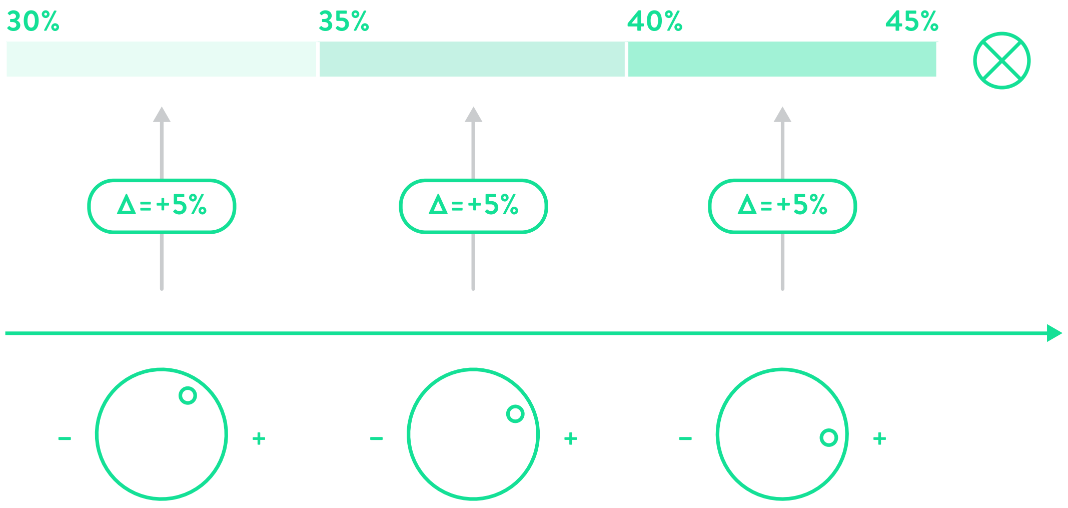
To make our lighting control system slightly more sophisticated, we'll add light level control capabilities. First, the specification allows us to decide what will happen once a light source is powered up. This behavior is determined by the Generic OnPowerUp state. In order to implement it, a light source requires the Generic Power OnOff Server model and the Generic Power OnOff Setup Server model. The former extends the Generic OnOff Server model, which means that the Generic OnOff Server model also needs to be implemented. This allows us to choose from one of the following modes:

1. **Off** – when a light source is powered up, its light level is automatically set to 0%
2. **Default** – when a light source is powered up, its light level is automatically set to a desired value
3. **Restore** – when a light source is powered up, it restores the state it was in when powered down

Second, the specification covers real-time light level control capabilities which enable the classic dimming functionality known from wired systems. Generic Level Server and Generic Level Client are the models that need to be implemented in order to enable light output adjustments. As in our previous examples, light sources must implement the server model, while dimmers rely on the client model. This is a universal rule that applies to all lighting control scenarios utilizing Bluetooth mesh connectivity. However, Generic Level models would not be enough to ensure smooth light level control within a range of 0% to 100%. Due to the nature of the human eye, there is a significant difference between measured and perceived light levels. To achieve perceived linear dimming, the intensity of light must be matched to the way our eyes work. This is what Light Lightness Server and Light Lightness Client models take care of. They introduce the Light Lightness Linear state and Light Lightness Actual state. The former represents the lightness on a linear scale, while 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 is what eventually ensures a smooth, human eye-friendly dimming experience. The binding between the Light Lightness Linear state and the Light Lightness Actual state means that whenever one of them changes, the other one 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 light intensity. The specification explains these relations thoroughly, even outlining how the scientific community's understanding of the exact relationship between the perceived lightness of a light and measured light intensity has changed over time (see: *Appendix A.2: Light in numbers*).

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

1. The light level can be **automatically adjusted to reach a desired value** within a range of 0% to 100%. This is a new absolute value, regardless of what the previous state of the lamp was.
2. The light level can be **adjusted by a specified Delta value** (e.g. +5%). This results in a relative change versus the previous state of a device. The process supports changing the state by a cumulative value with a sequence of messages that are part of a transaction. This means that a dimmer can be used to perform multiple Delta adjustments that will be sequenced into one cumulative operation, as shown in the diagram below:



‹ **Figure 5. Standard dimming**

Quick recap:

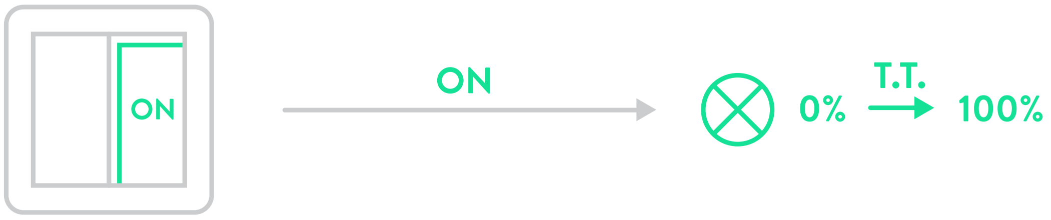
nodes involved	light sources	dimmers
models used	Generic Power OnOff Server	
	Generic Power OnOff Setup Server	
	Generic OnOff Server	Generic OnOff Client
	Generic Level Server	Generic Level Client
	Light Lightness Server	Light Lightness Client
paradigm	subscribe	publish

Transitions

In the previous section, we learned how to control the light level using a dimmer. But whenever a dimmer requests a light source to change its light level from one value to another (e.g. when a new absolute light level value is requested), the question remains how long this process should take. The time it takes for a state to change from a present state to the target state is called the transition time. It can be adjusted very flexibly using the Generic Default Transition Time Server model. It covers a wide range of times, from milliseconds to hours, and is thus capable of handling a variety of applications. Once implemented, the model imposes the following procedure:

- If the transition time isn't specified in a state-changing message, then use the default transition time specified for this device
- If the transition time is specified in a state-changing message, apply it

The diagram below illustrates the concept of transition times using an example of a light source that has been turned on (100% output). Once the message arrives, the light source will be increasing its light level from 0% to 100% over the specified transition time (T.T.).



‹ **Figure 6. Transition times**

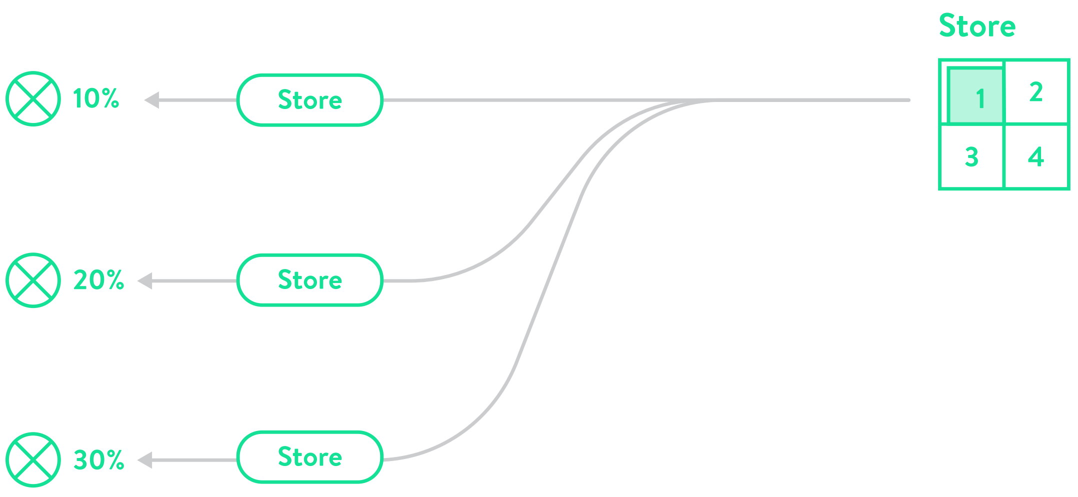
Quick recap

nodes involved	light sources	configuration devices
models used	Generic Default Transition Time Server	Generic Default Transition Time Client

Scenes

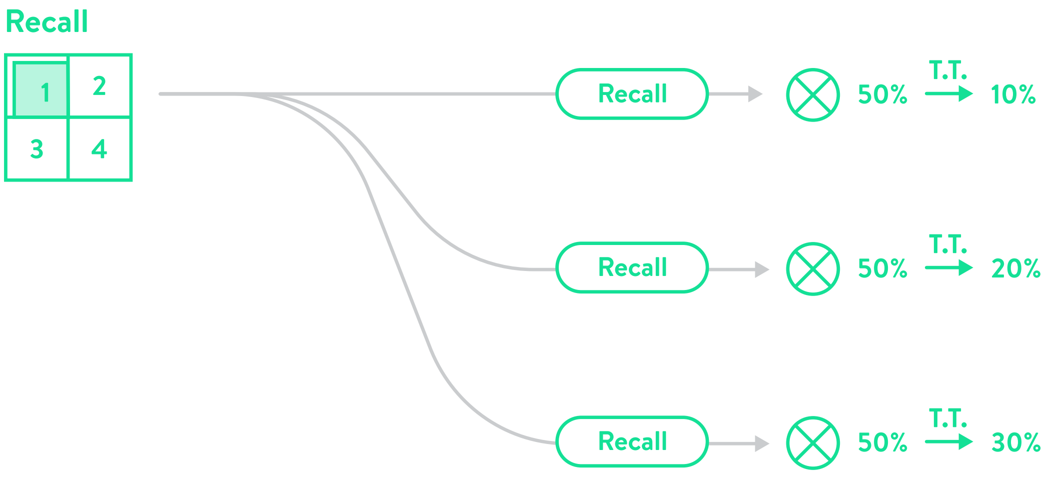
The specification introduces a mechanism which allows for memorizing states of devices (e.g. light level or color) so that they can be restored on demand or based on a preset schedule. On the light source side, scenes are handled by two models. The first one is the Scene Setup Server model which takes care of scene configuration, making it possible to memorize desired states of devices. The second one is the Scene Server model which is used to restore the previously saved scenes. On the switch side, scenes are handled by the Scene Client model.

Let's say we want to create a scene involving three light sources, each with a different light level: 10%, 20% and 30%. To do this, we need to "manually" set their outputs to reflect desired values. This can be done using methods described in the section "Light level control" above. Then, the Scene Store operation needs to be performed for each of these lamps. As part of this process, a message is sent to the Scene Setup Server model in each lamp, requesting it to memorize the state of the lamp under a selected Scene Number. The Scene Number is an 16-bit network-wide value, which means that a mesh network can accommodate up to 65,535 scenes. If a light source is in the middle of a transition process when the Scene Store message arrives, then its target state will be stored for that particular scene. The entire process can be illustrated as follows:



‹ **Figure 7.**
The Scene Store operation

Once the Scene Store operation is performed for all of our three light sources, we can recall our scene whenever needed by referencing the Scene Number. This operation is called Scene Recall and it applies the stored values of states to all lamps included in a given scene. If the Scene Recall message arrives when a lamp is in the middle of transition, then the transition process will be interrupted. In our example, the Scene Recall operation would mean setting our lamps' light levels at 10%, 20% and 30%, respectively, regardless of what their previous states were. This is shown in the diagram below:



‹ **Figure 8.**
The Scene Recall operation

The transition time (TT) for a given scene can be specified in the Scene Recall message, which will make all of our light sources reach their previously saved light levels over the same period of time. Scene Store and Scene Recall operations are handled by different models, which is important from the network operation perspective as it allows for separating network configuration from device control. Having the key to a particular model, network administrators can define and store desired scenes, while end users can only recall them.

Quick recap

nodes involved	light sources	switches/dimmers	configuration devices
models used	Scene Setup Server	Scene Client	Scene Client
	Scene Server		

Sensors

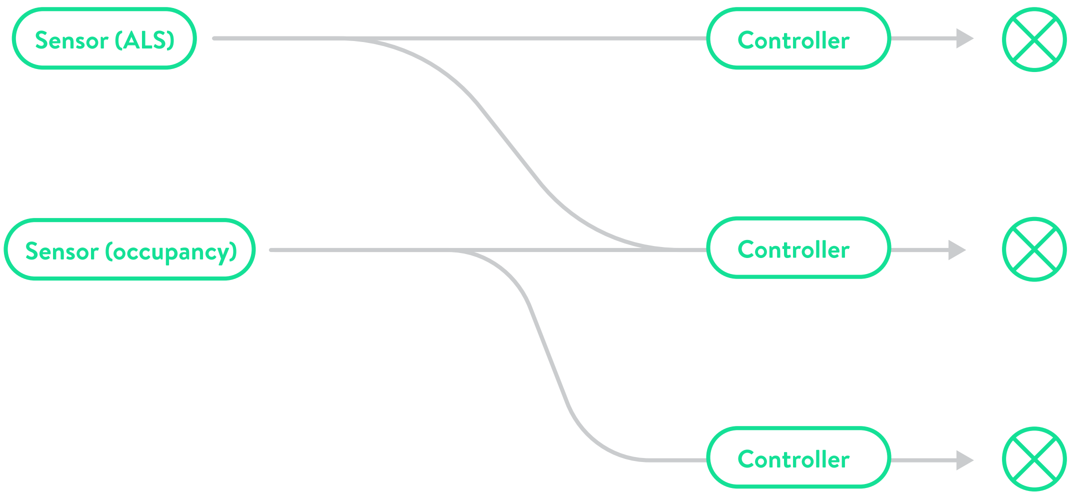
It's time to make our installation aware of what is going on inside the space, and to grant luminaires a certain degree of autonomy. We'll achieve this by adding occupancy and ambient light sensors. A sensor-driven connected lighting network is capable of tracking changes in the environment and interpreting these changes in order to dynamically adjust its operation for optimal comfort and energy efficiency, as well as lower electricity bills. Advanced lighting control strategies based on occupancy and ambient light sensors also enable commercial buildings to meet the requirements of building energy codes, such as California's Title 24 standards. The Bluetooth mesh networking specification includes mesh models that fully support both occupancy sensing and daylight harvesting. They come with multiple tunable parameters and properties, effectively future-proofing buildings against more stringent environmental requirements that we can expect to see as the world strives for a low carbon future.

In the Bluetooth mesh networking specification, sensor is a device which only publishes information. Whether we talk about temperature, humidity or ambient light level sensors, they all do basically the same job - report the current state of things. Occupancy sensors are pretty straightforward devices. Even though they might be able to report some additional data (the specification supports such properties as Motion Sensed Property or Time Since Motion Sensed), usually we'll want them to report one of two values: occupancy detected / no occupancy detected. Ambient light sensors are a different story, reporting illuminance measurements from a vast range of 0.00 to 167,772.16 lux (with an accuracy of 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/dimmers are designed for humans, allowing them to directly control the lighting environment. Sensors do not require human interaction. They automatically publish information upon which lamps adjust their output. The frequency of such reporting can be adjusted depending on how often we want a particular sensor to check and publish relevant information. However, in certain cases regular reporting is not sufficient to achieve desired results. This is why the specification allows for implementing a procedure that increases the frequency of reporting when sensor reading exceeds certain predefined threshold (lower or upper), or when the rate of change of the sensor reading exceeds certain predefined value. Sensor models outlined in the specification also have multiple configuration properties. All of this gives enormous design flexibility which allows connected lighting installations to be easily 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. This means that both occupancy sensors and ambient light sensors need to implement these models. Once we have a sensory infrastructure in place, we need to implement certain rules that will allow specific behaviors to be triggered by sensor readings. This is what the controller is all about.

Controller

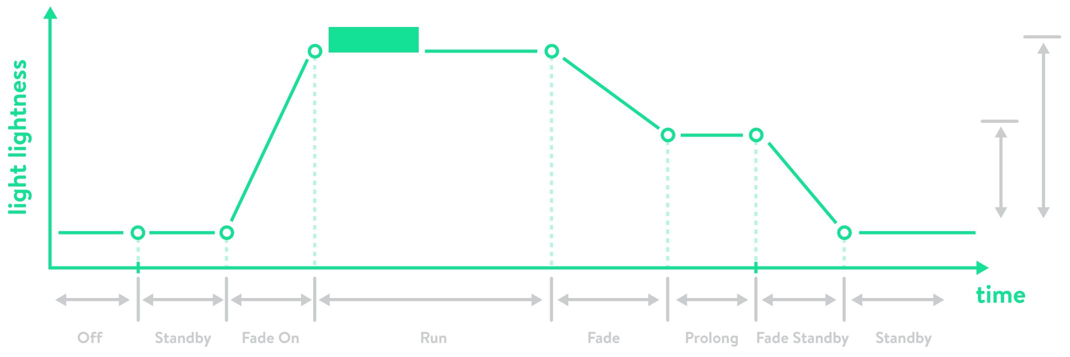
The Light Lightness controller controls the lamp's light level based on information obtained from occupancy and ambient light level sensors. It is a piece of software that sits inside each smart luminaire that belongs to a connected lighting network based on Bluetooth mesh. In such networks, there is no need for any dedicated controller boxes. The architecture is decentralized and doesn't include single points of failure. It is the software that provides all relevant capabilities and ensures desired operation of the entire installation.



‹ **Figure 9. Peer-to-peer topology of a Bluetooth mesh lighting network: sensors publishing to groups of LEDs with built-in software controllers**

SILVAIR

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



‹ **Figure 10. Operation of a Light Lightness controller based on occupancy sensor readings**

Starting from the left side of the graph, we can see that the light level increases gradually once a sensor detects movement in a given area. The thick line represents the period during which movement is still being detected. The light level remains unchanged for a specified period of time after the last sensor reading reflecting a detected movement. Then it gradually transitions to the so-called Prolong phase. Again, this phase lasts for a specified period of time after which the controller gradually returns to the Standby phase. All of the double-headed arrows represent properties 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, while vertical ones represent settings adjusting desired light levels. These levels can be hard-specified by the administrator, but can also be stabilized with the help of ambient light level sensors. The calculations used as part of this lightness stabilization process are included in the Light LC Proportional-Integral Feedback Regulator. The specification also allows any adaptive control strategies to be overridden by end users whenever needed.

Quick recap

nodes involved	light sources + controller	sensors
models used	Sensor Client	Sensor Server
		Sensor Setup Server
	Light LC Server	
	Light Lightness Server	
	Light LC Setup Server	
paradigm	subscribe	publish

In addition to single-function sensors reporting only specific data, there are also multisensors that are capable of providing several different types of readings. A multisensor can, for example, measure occupancy, ambient light level and temperature, and send all this information in a single message. This provides clear benefits in terms of energy efficiency and overall performance simplicity, but this is also what makes multisensors quite challenging devices from the point of view of mesh networks. Before we get to details, we'll first need to get familiar with the concept of elements - one of the most important architectural concepts outlined in the Bluetooth mesh networking specification.

Elements and multisensors

As the specification puts it, *an element is an addressable entity within a node*. There are many examples of mesh network nodes that have only one element (it is called the **primary element**) and wouldn't even require the concept of elements in order to function properly. However, devices like multi-lamp fixtures need to utilize this concept in order to reach their full potential in a mesh lighting network. In addition to the primary element, such nodes have one or more **secondary elements.**

Multisensors are typically single-element devices. This results from their architecture: even though they contain a number of different sensors, they report all these measurements in a single transmission. It is not possible to identify separate addressable entities within them, so they cannot employ the multi-element concept from the Bluetooth mesh networking specification. Therefore, if we take our example of a multisensor measuring occupancy, ambient light level and temperature, it turns out that such device cannot be used to feed the Light Lightness controller with the data it needs to implement occupancy sensing and daylight harvesting at the same time. These functionalities can't be grouped independently without a multi-element concept implemented.

This brings us to the conclusion that multisensors need to include multiple elements - so that sensor readings affecting similar functionalities (e.g. occupancy readings and ambient light readings) can be sent in separate messages. In the case of our multisensor, things would work perfectly fine if the occupancy measurement was transmitted in one message, while the ambient light measurement together with the temperature measurement would travel via another message. Such an approach is slightly less energy efficient, but it allows for independent grouping of network nodes, while also allowing different reporting frequency for occupancy and ambient light readings. This is something that manufacturers need to take into account when designing mesh-ready sensor devices.

Building a complete installation

At this point we have all components we need to build an adaptive, sensor-driven connected lighting system. In order to set up such installation, we'll need to make sure that each network node implements required mesh models as described in all sections above. In addition to Foundation Models mentioned in the introduction, individual groups of devices need to implement the following models:

light sources	**switches / dimmers**	**sensors**	**configuration devices**
Generic OnOff Server	Generic OnOff Client		
Generic Power OnOff Server			
Generic Power OnOff Setup Server			Generic Power OnOff Client
Generic Level Server	Generic Level Client		
Generic Default Transition Time Server			Generic Default Transition Time Client
Scene Setup Server			Scene Client

Scene Server	Scene Client		
		Sensor Server	
		Sensor Setup Server	Sensor Client
Light Lightness Server	Light Lightness Client		
Light Lightness Setup Server			Light Lightness Client
Light LC Server			Light LC Client
Light LC Setup Server			Light LC Client

Before such network can operate in accordance with specified requirements (e.g. applicable building energy codes), relevant properties - such as transition times, frequencies of sensor readings, duration of controller phases or required light levels - must be adjusted to match desired values.

In practice, some sort of a **provisioner** is also required to configure a mesh lighting network. This can be a smartphone with an app that allows for implementing relevant models, setting up groups and scenes, and adjusting relevant settings in a user-friendly way. A cloud platform is also needed in order to safely store the network configuration backup and allow for expanding a configured network in the future. Developing efficient, intuitive and user-friendly commissioning and maintenance tools is a challenging task, as design of these tools is going to have a huge impact on the overall connected lighting experience. The Bluetooth mesh networking specification lays ground for unprecedentedly robust and scalable smart lighting installations, but the stage of specification implementation is where this enormous potential can be fully unleashed or wasted.

The scope of functionalities described in this paper represents only part of possibilities enabled by mesh models included in the specification. The document also addresses e.g. **time synchronization,** allowing implementation of precise **time scheduling.** The more sophisticated system we want to build, the more mesh models it will require - but the instructions and core paradigms described above should be a good starting point for building complex mesh installations.

About us

Silvair provides complete and flexible lighting control solutions based on Bluetooth mesh networking technology. Component manufacturers can easily integrate them into a variety of products, becoming part of a globally interoperable ecosystem. In addition, our intelligent lighting platform includes a set of dedicated tools for commissioning and managing connected lighting systems in commercial spaces.

We give our partners a head start with a ready-to-use wireless technology, the shortest time to market, and guaranteed compliance with the Bluetooth mesh specification.

For more information visit www.silvair.com

If you have any questions, contact us at: business@silvair.com

For media inquiries, please contact: media@silvair.com

Our offices:

Europe
Jasnogorska 44
31-358 Krakow
POLAND

North America
717 Market Street, Suite 100
San Francisco, CA 94103
USA

SILVAIR

SILVAIR