Introduction to Z-wave and OpenZWave

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

MSE | MASTER OF SCIENCE IN ENGINEERING

Plan

- Z-Wave
- Python OpenZWave

Z-wave

- Z-Wave is an international standard for wireless home automation.
- Home automation allows to interconnect all functions dealing with electricity such as light, heating, cooking, cooling, security etc. with each other and to apply automation of these functions.
- Z-Wave uses low-powered radio waves.
  - Its mesh network covers all areas of the home, as the radio waves travel easily through walls, floors and furniture, making connectivity 100% reliable.

Requirements of a wireless system for home control

- Reliability of the communication
- Security of communication
- Low radio emission
- Simple usage
- Adequate price
- Protection of investment
- Interoperability

For more details, read “Z-Wave Technical Basics” document (sections 1.1 and 1.2)

Z-wave history

- End of the nineties: Zen-Sys (Dinesh company) invented Z-Wave
- From the initial idea of developing their own home automation solution, the company soon evolved into becoming a chip provider selling a home automation ASIC.
  - Ecosystem of manufacturers with compatible products.

Z-wave history

- The first generation of Zensys hardware was sold in 2003
- 4 generations : 2003, 2005, 2007, 2009
- 2007: The first larger Z-Wave device manufacturer in Europe was the German switch manufacturer Merten (now a part of Schneider Electric).
- 2008: Sigma Design (Asian manufacturer) bought Zen-Sys
- 2009: Z-Wave gets adopters in Asia.

Z-wave alliance today

- 600 companies*
- 2100 products*
- All interoperable with each other

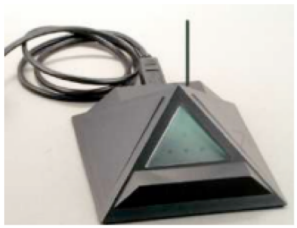
*: https://z-wavealliance.org/

General Layer model

- Application layer
- Network layer
- Radio layer

Z-wave devices

- Z-Wave distinguishes two basic types from devices:
  - Controllers are Z-Wave devices that can control other Z-Wave devices,
  - Slaves are Z-Wave devices that are controlled by other Z-Wave devices.



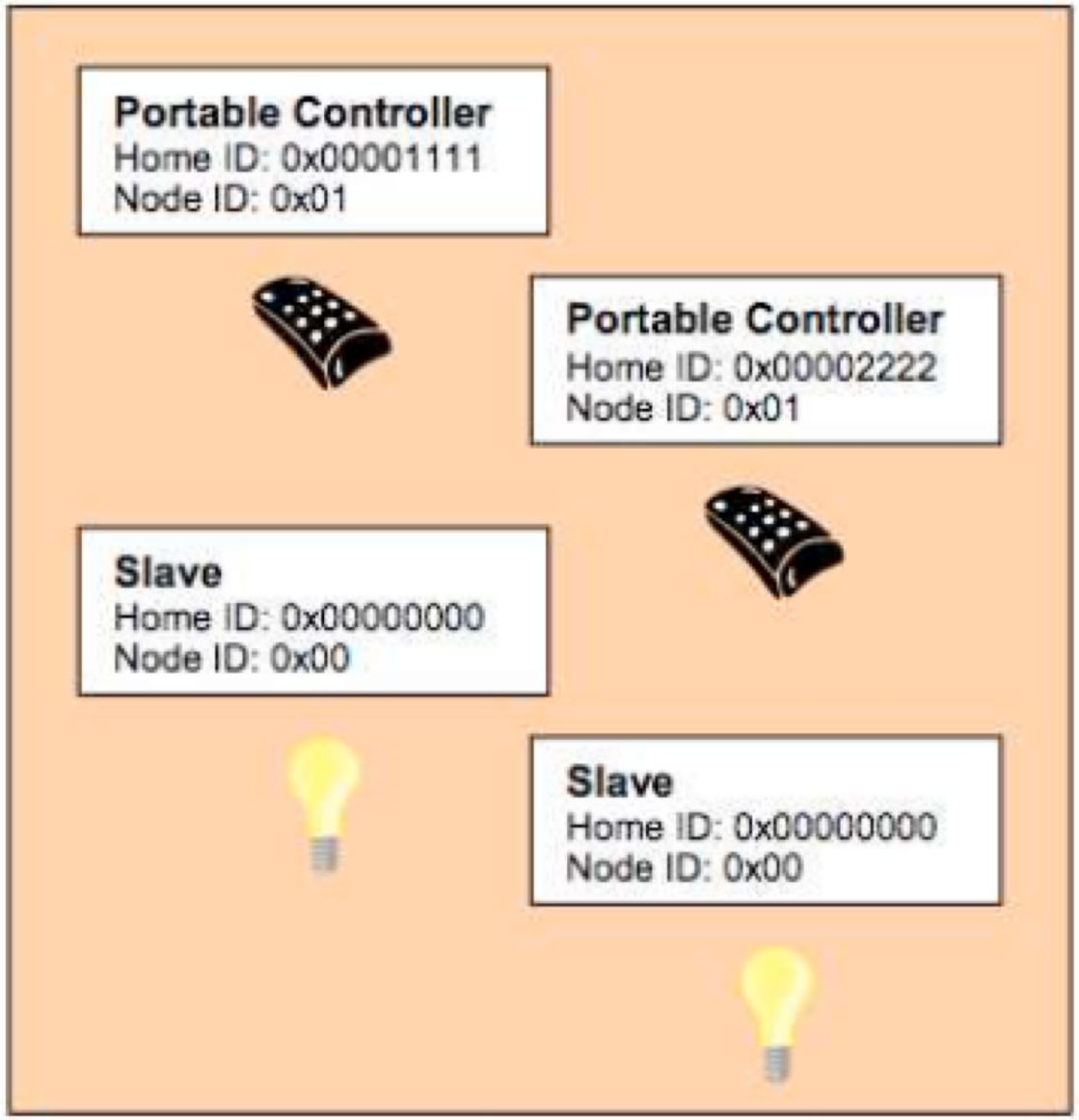
Z-wave devices

- Each Z-Wave module can act as a repeater and commands can route through a maximum of four devices. This gives the system a maximum range of ~ 120 m
- Routing is managed automatically

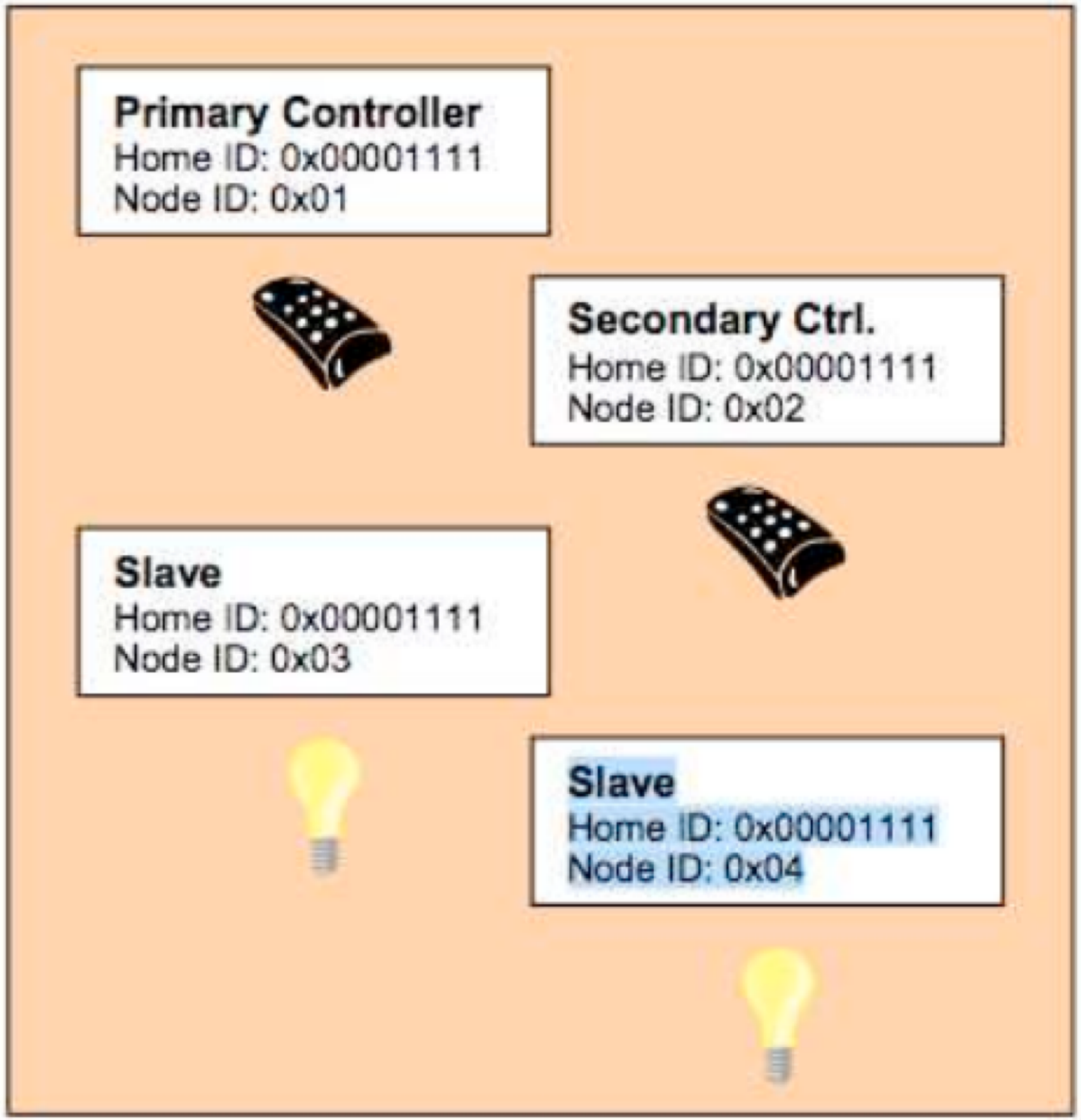
Z-wave identifications

- The Z-Wave protocol defines two identifications for the organisation of the network:
  - The Home ID is the common identification of all nodes belonging to one logical Z-Wave network. It has a length of 4 bytes = 32 bits and is less of interest for the final user.
  - The Node ID is the address of the single node in the network. The Node ID has a length of 1 byte = 8 bits.

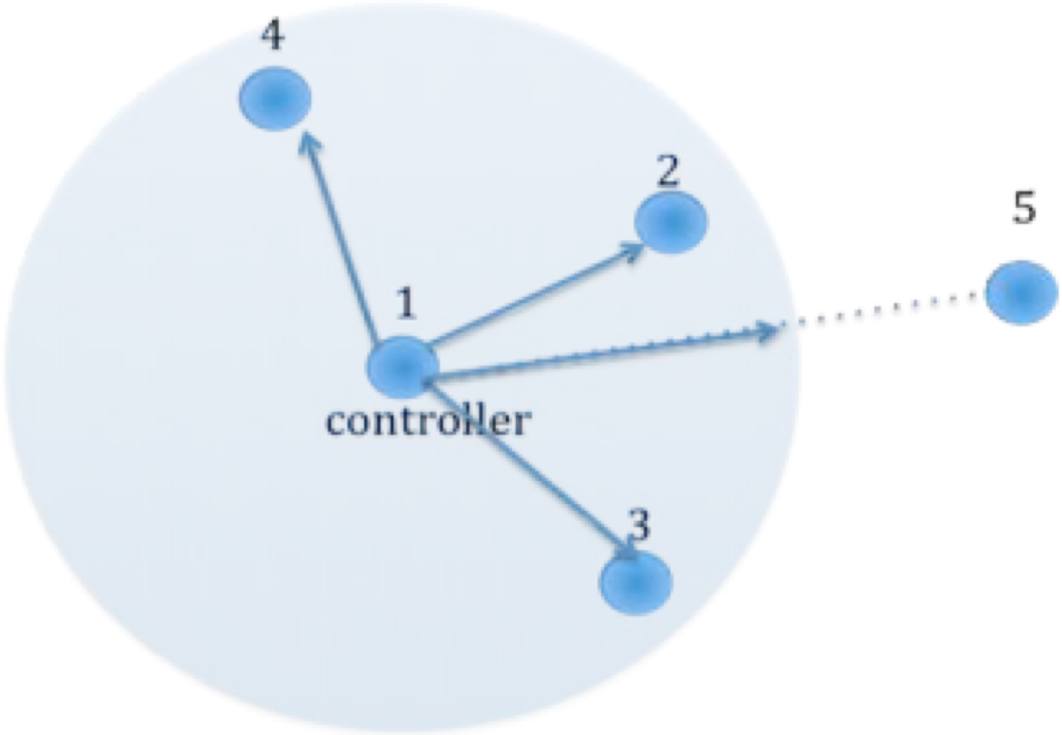
Z-wave : Before inclusion



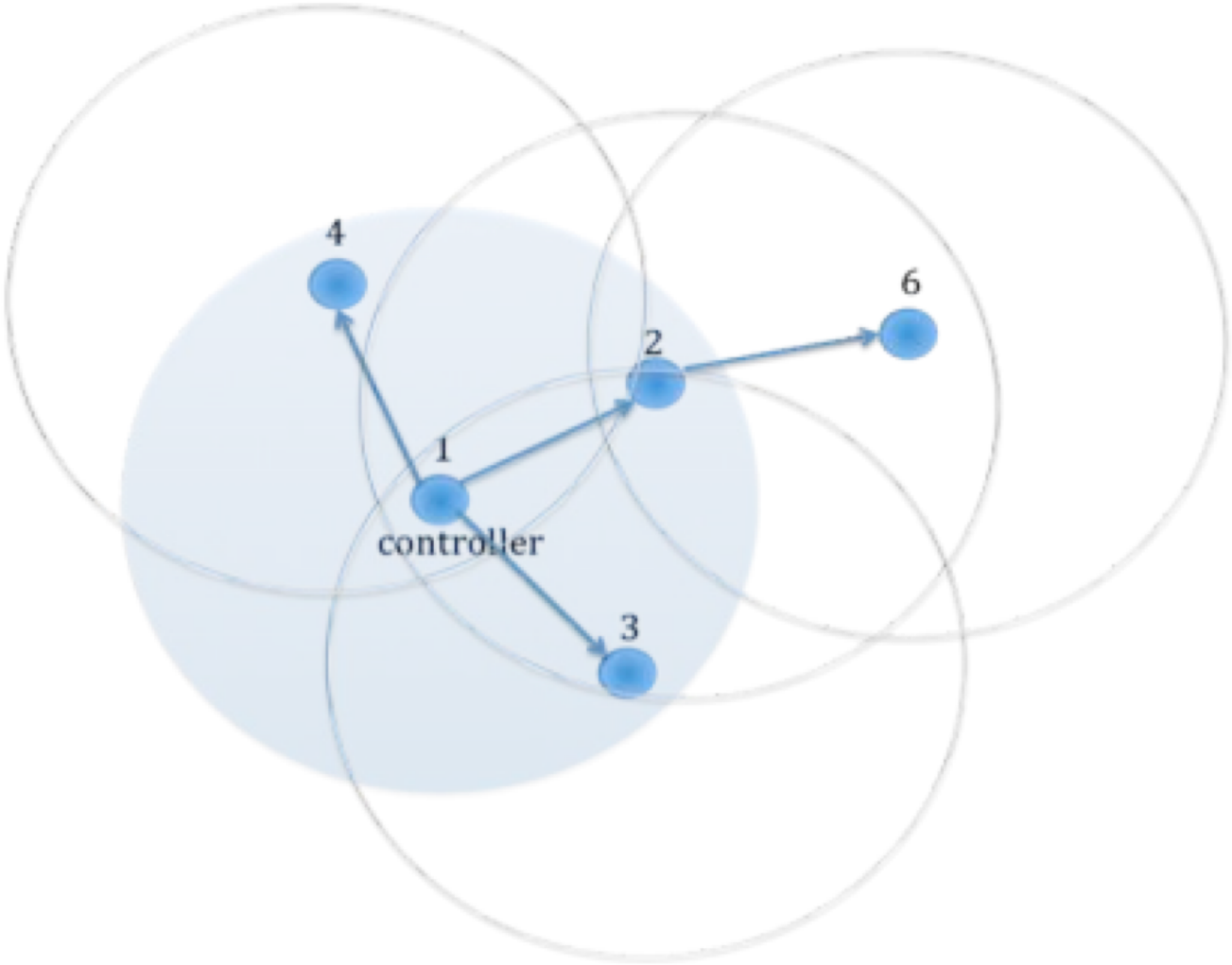
Z-wave : After inclusion



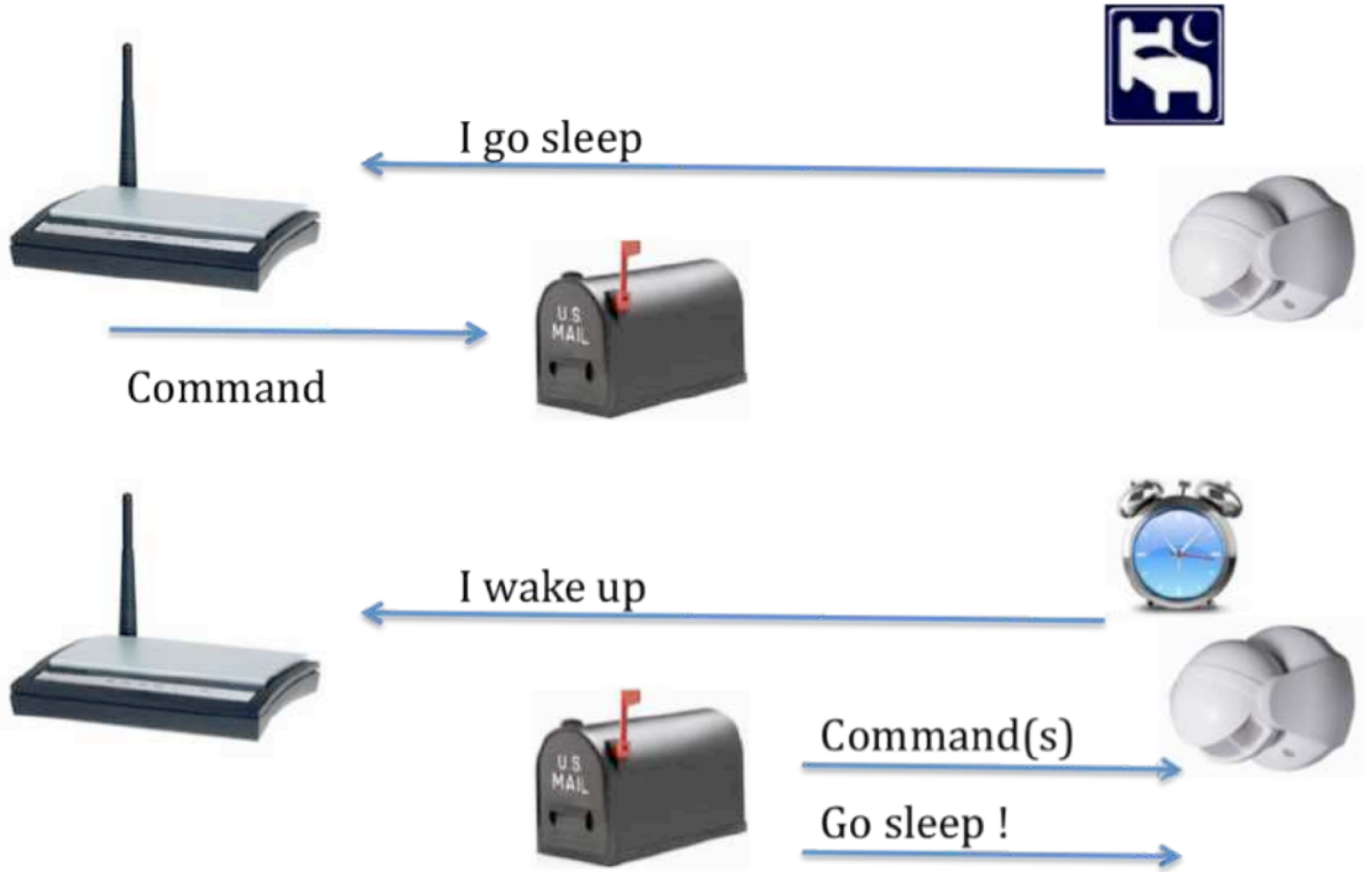
Z-wave network (without routing)



Z-wave network (with routing)



Battery operated devices



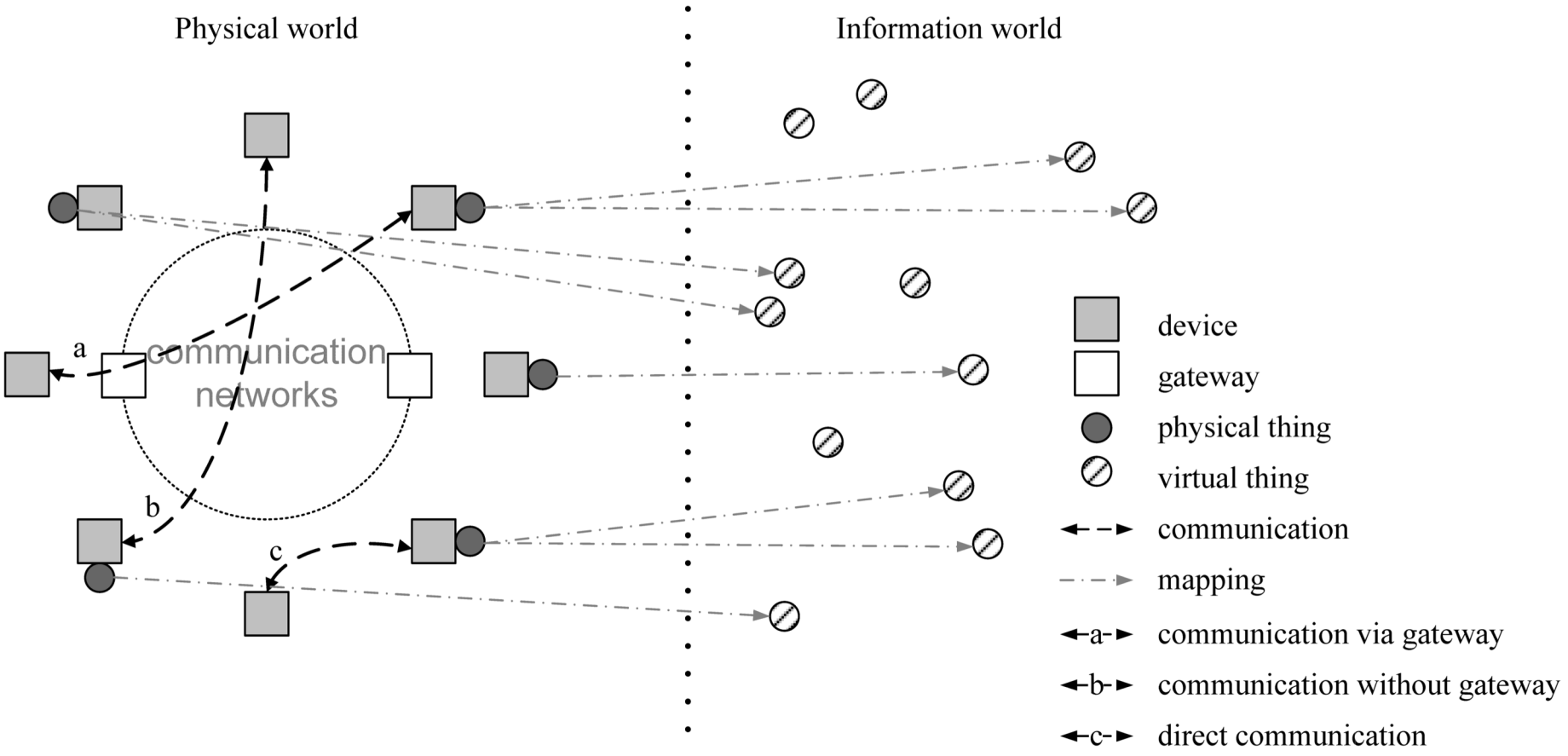
“Z-Wave Technical Basics” document

- 125 pages document
- Introduction : p 6 – p 16
  - Comparison of proprietary protocols
  - History and Characteristics of Z-Wave
- Network Layer: p 26 – p 63

Plan

- Z-Wave
- Python OpenZWave

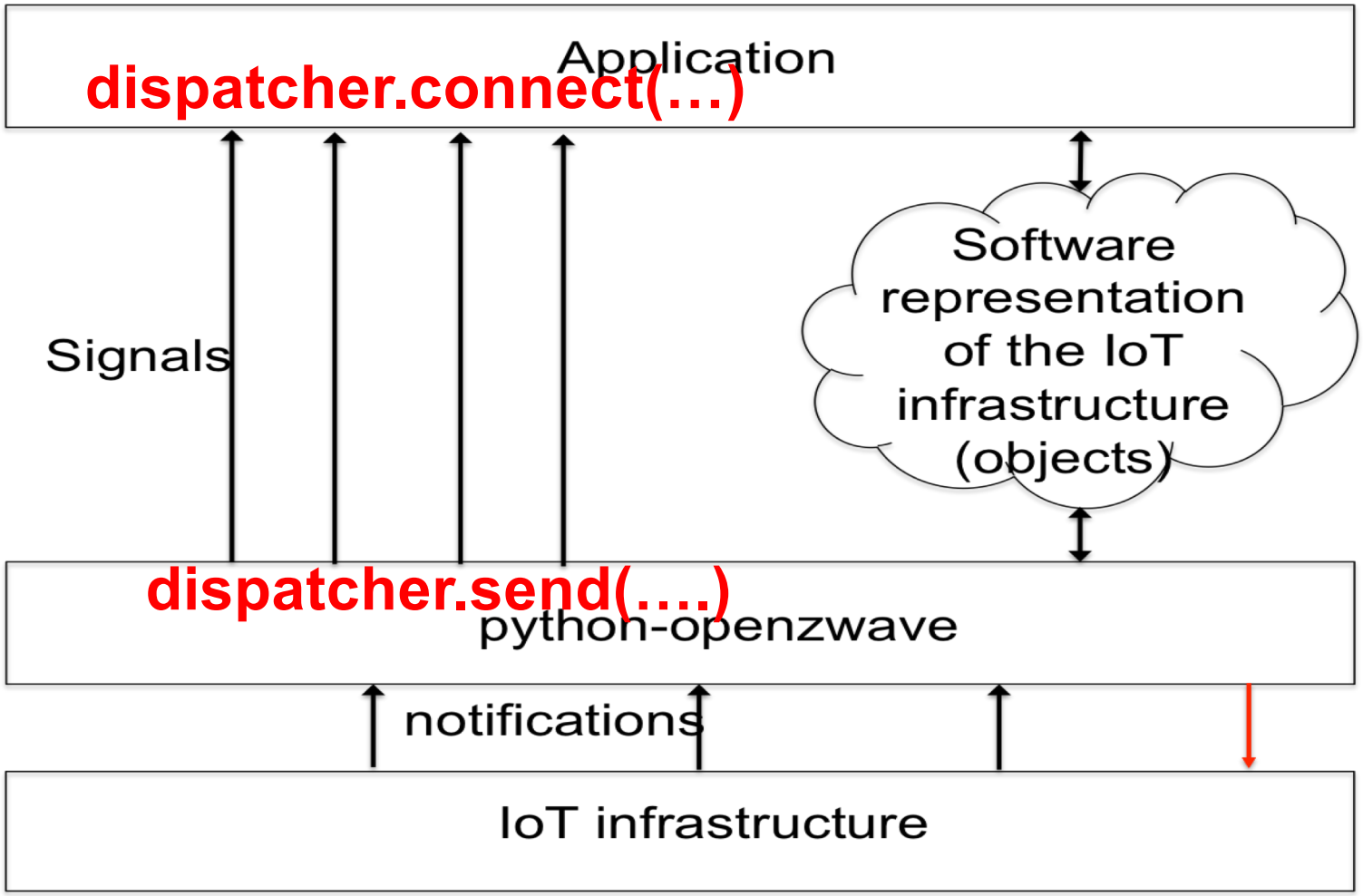
Reminder …


Physical world
Information world
communication networks
a
b
c
device
gateway
physical thing
virtual thing
communication
mapping
a communication via gateway
b communication without gateway
c direct communication

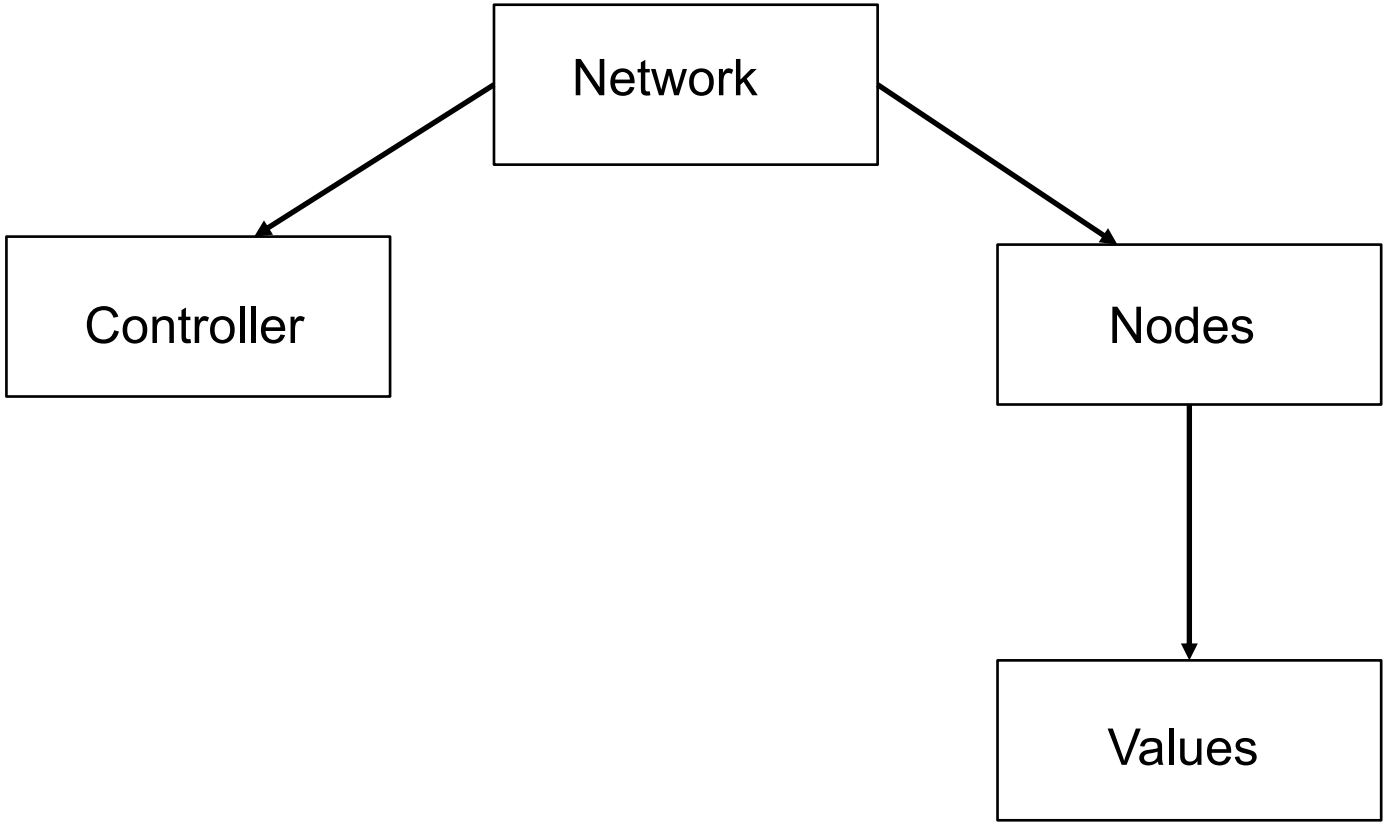
Z-Wave

- OpenZWave is an open-source, cross-platform library designed to enable anyone to add support for Z-Wave devices to their applications, without requiring any in depth knowledge of the Z-Wave protocol.

python-openzwave



Virtual representation



Network class

- **home_id_str**
- **is_ready** : (type: Boolean): indicates if the network is ready or not. As soon as the network is ready, a *SIGNAL_NETWORK_READY* event is generated by python-openzwave.
- **nodes_count** : (including the controller)
- **controller** : (type:Controller)
- **nodes** : a dictionary of "node" objects representing the nodes connected to the network (including the controller).
- **start ()** : This method generates the software representation of the Z-Wave network managed by the controller.
- **stop ()** : this method deletes the software representation

Controller class

- **name**: controller's name.
- **device**: path of the device's driver (e.g. /dev/USB0)
- **begin_command_add_device()** (return type: Boolean): gets the controller in inclusion mode. The method returns "True" if the command is accepted and started.
- **begin_command_remove_device()** (return type: Boolean): gets the controller in exclusion mode. The method returns "True" if the command is accepted and started.
- **cancel_command()** (return type:none): gets the controller out of the inclusion or exclusion mode.

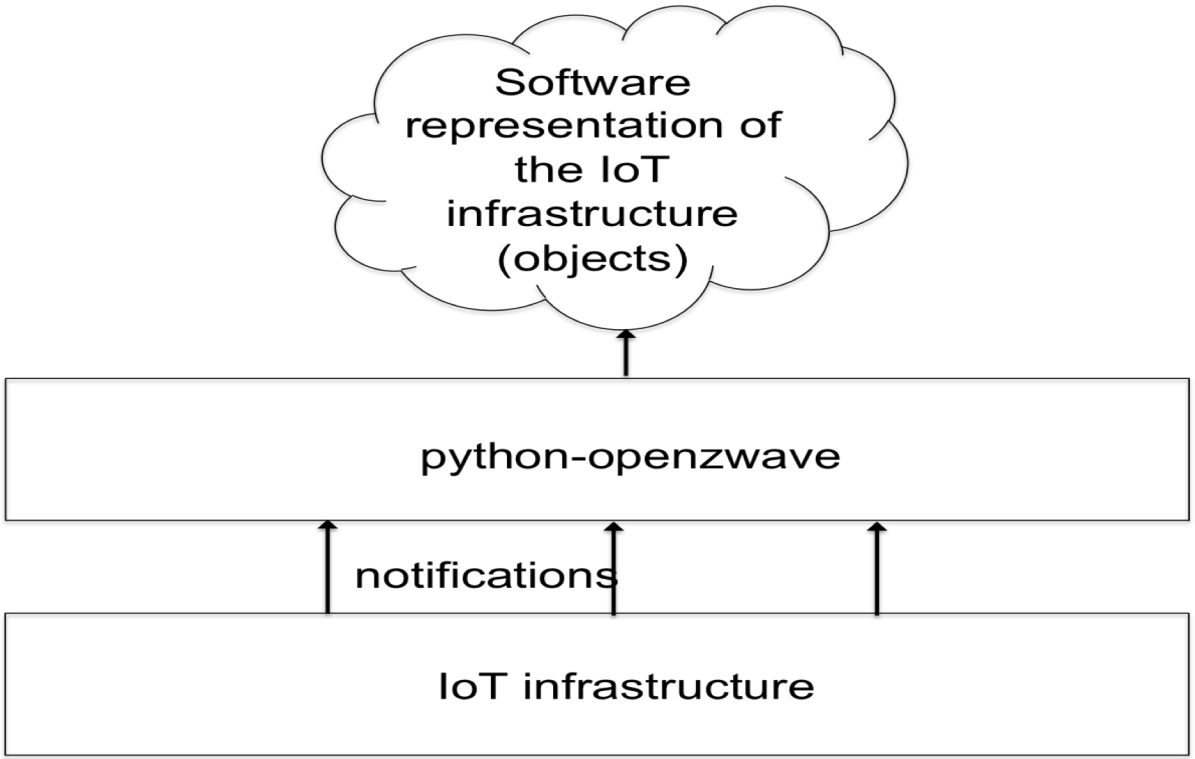
Node class

- **node_id** (type: integer): identifying the node in the network (note: the node with the id "1" is always the controller).
- **product_name** (type:string): name of the product. (Sensor's product name is "Multisensor 6" and dimmer's product name is "ZE27")
- **name** (type:string): name of the node.
- **location** (type:string) : location of the node.
- **isReady** (type: Boolean): Indicates whether the node is ready or not (this method is not in the documentation) -→ deprecated (is_ready)
- **getNodeQueryStage** (type:String): Indicates the State of the Node object (Once the Query Stage is at "Complete", this Node object is ready to be used).
- **neighbors** (type: set()) : a set of the neighbouring nodes. Neighbouring nodes are nodes that can be reached by the current node
- **values** (type:dict()): This is a dictionary containing pairs of (ID, value_object). A “value” object represents a measure or a configuration parameter of this Node.

Value class

- **label** (type:string): the label of the value (eg. "Temperature" in the case of temperature value recovered by the sensor node or "Level" in case of brightness level).
- **data** (type:depends on the type of the value) : value of the measurement.
- **units** (type:string) : unit of the value

python-openzwave



python-openzwave