Software Design Specification

Z-Wave Protocol Overview

Document No.:	SDS10243
Version:	8
Description:	This document is a high-level description of the Z Wave Protocol.
Written By:	JFR;PSH;ABR;JBU
Date:	2008-12-04
Reviewed By:	CHL;JKA
Restrictions:	Partners Only

Approved by:

Date	CET	Initials	Name	Justification
2008-12-04	13:49:23	JFR	Jørgen Franck	on behalf of NTJ

CONFIDENTIAL

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20050128	JFR	-	Removed paragraph about assignment of home and node ID Removed paragraph about controller replication
2	20060105	MVO	All	New 1st page/header/footer contents. New Doc No
3	20070228	JFR	3.4 All	Frequently listening routing slaves and Zensor Net routing slaves added Table of figures added
4	20070509	JFR	All	Confidential in footer removed
5	20080626	ABR	3.5	Random home ID after reset in controllers and slaves added
6	20081124	JFR	3.4.2 & 3.4.4	ZDK 5.0x supports FLiRS and Zensor Net only.
7	20081126	JFR	All	Minor typos corrected
8	20081202	ABR	5.1.4.1 & 6.4	Explorer frame type and explorer route resolution added

Table of Contents

List of Figures

1 ABBREVIATIONS

Abbreviation	Explanation
EOF	End Of Frame
FLiRS	Frequently Listening Routing Slave
MAC	Media Access Control
PIR	Passive Infra Red movement sensor
RF	Radio Frequency
SIS	SUC ID Server
SOF	Start Of Frame
SUC	Static Update Controller
TTL	Time-To-Live
ZDK	Z-Wave Developer's Kit

2 INTRODUCTION

2.1 Purpose

The purpose of this document is to describe the Z-Wave^{TM} Radio Frequency Protocol that provides reliable and robust wireless communication between the nodes in a Z-Wave mesh network.

2.2 Scope

The scope of this document is to give an overview of the following protocol layers:

- The MAC layer
- The Transfer Layer
- The Routing Layer
- The Frame Layer

2.3 Audience and Prerequisites

The audience of this document is Z-Wave partners and Zensys A/S.

3 Z-WAVE PROTOCOL

3.1 Overview

The Z-Wave protocol is a low bandwidth half-duplex protocol designed for reliable and robust wireless communication in a low cost control mesh network. The protocols main purpose is to communicate short control messages in a reliable manner from a control unit to one or more nodes in the network.

The protocol is not designed to transfer large amounts of data or to transfer any kind of streaming or timing critical data.

The protocol consist of 4 layers, the MAC Layer that controls the RF media, the Transfer Layer that handles frame integrity checks, acknowledgements, and retransmissions, the Routing Layer that controls the routing of frames in the network and application interface; and finally the Application Layer controls the payload in the transmitted and received frames.

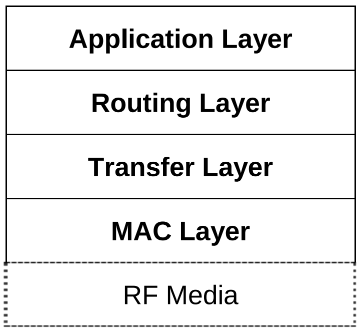


Figure 1 Z-Wave Protocol Layers

For a description of the four layers, refer to chapters 4 to 7.

3.2 Controller and Slave nodes

The Z-Wave protocol has two basic kinds of devices; controlling devices and slave nodes. Controlling devices are the nodes in a network that initiate control commands and sends out the commands to other nodes, and slave nodes are the nodes that reply on and execute the commands. Slave nodes can also forward commands to other nodes, which make it possible for the controller to communicate with nodes out of the direct radio wave reach.

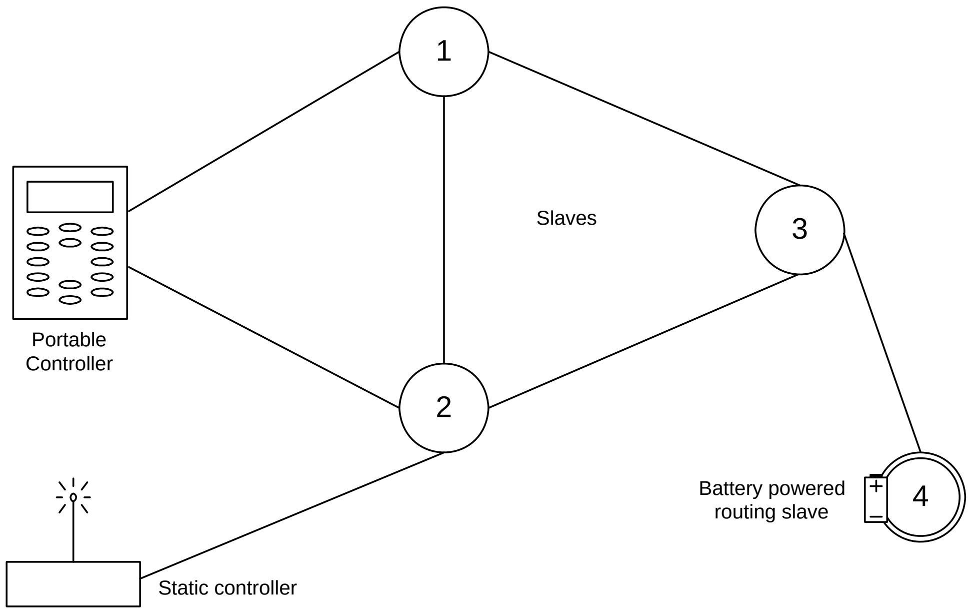


Figure 2 Z-Wave network comprising of controllers and slaves

3.3 Controllers

A controller is a Z-Wave device that has a full routing table and is therefore able to communicate with all nodes in the Z-Wave network. The functionality available in a controller depends on when it entered the Z-Wave network. In case the controller is used to create a new Z-Wave network it automatically become the primary controller. The primary controller is the “master” controller in the Z-Wave network and there can only be one in each network. Only primary controllers have the capability to include/exclude nodes in the network and therefore always have the latest network topology.

Controllers added to the network using the primary controller are called secondary controllers and don’t have the capability to include/exclude nodes in the network.

3.3.1 Portable Controller

A portable controller is a controller, which is designed to change position in the Z-Wave network. The portable controller uses a number of mechanisms to estimate the current location and hereby calculating the fastest route through the network. A portable controller is typically battery powered because it is mainly used in portable applications. It is difficult to communicate with a portable controller because it is typically powered down when not in active use to prolong battery lifetime.

An example of a portable controller could be a remote control.

3.3.2 Static Controller

A static controller is a fixed controller that must not change position in the network and has to be powered up all the time (always listening). This controller has the advantage that Routing slaves can report unsolicited status messages to it, and it also has the advantage of always knowing where it is located in the network. A static controller will typically be a secondary controller in a Z-Wave network.

An example of a static controller could be an Internet gateway that monitors a Z-Wave system.

3.3.2.1 Static Update Controller

A Z-Wave network can optionally have a static controller with enabled Static Update Controller (SUC) functionality to distribute network topology updates. A SUC is a static controller that will receive notifications from the primary controller regarding all changes made to the network topology. In addition, the SUC is capable of sending network topology updates to other controllers and routing slaves upon request. The primary controller assigns the SUC role to a static controller from application level. The static controller will only accept the SUC role in case it has enabled assignment. There can only be one SUC in a Z-Wave network.

3.3.2.2 SUC ID Server

A Z-Wave network can optionally have a SUC with enabled node ID server functionality (SIS). The SIS enables other controllers to include/exclude nodes in the network on its behalf. The SIS is the primary controller in the network because it has the latest update of the network topology and capability to include/exclude nodes in the network. When including additional controllers to the network they become inclusion controllers because they have the capability to include/exclude nodes in the network on behalf of the SIS. The inclusion controller’s network topology origin from last time a node was included or it requested a network update from the SIS. Categorizing the inclusion controller as a primary controller is therefore wrong because its network topology may be outdated.

3.3.3 Installer Controller

An Installer controller is a portable controller that has additional functionality, which enables it to do more sophisticated network management and network quality testing than other controllers.

An example of an installer controller could be an installation tool used by an installer to install a Z-Wave network at a customer site.

3.3.4 Bridge Controller

A Z-Wave network can optionally have a bridge controller. A bridge controller is an extended static controller, which incorporates extra functionality that can be used to implement controllers, targeted for bridging between the Z-Wave network and other networks. The bridge controller device stores the information concerning the nodes in the Z-Wave network and in addition it can control up to 128 virtual slave nodes. A virtual slave node is a slave node that corresponds to a node, which resides on a different network type.

An example of a bridge controller could be a bridge between an UPnP network and a Z-Wave network to link broadband and narrowband devices together in a home entertainment application.

3.4 Slaves

Slaves are a Z-Wave device with none or very limited knowledge about the network topology, and do not have the capability to include/exclude nodes to a Z-Wave network.

3.4.1 Slave

Slave nodes are nodes in a Z-Wave network that receives commands and performs an action based on the command. Slave nodes are unable to send routed messages to other slaves or controllers unless they are requested to do so in a command. Slave nodes acts as routers in the mesh network. A slave must be mains powered (always listening) to be able to receive commands from othe devices in the network.

An example of a slave node could be a light dimmer.

3.4.2 Routing Slave

Routing slaves has the same overall functionality as a slave. The major difference is that a routing slave can send unsolicited routed messages to other nodes in the network. They store a number of static routes for use when sending unsolicited messages to a limited number of nodes. A routing slave can be either mains or battery powered depending on the application and availability of mains power. Routing slave nodes acts only as routers in the mesh network when mains powered (always listening).

A special case of a battery powered routing slave is the Frequently Listening Routing Slave (FLiRS). This is a normal routing slave configured to listen for a wakeup beam in every wake-up interval. This enables other nodes to wakeup the FLiRS and sends a message to it. Notice that only ZDK 5.0x supports FLiRS nodes.

An example of a routing slave node could be a thermostat or a Passive Infra Red (PIR) movement sensor. One usage for a FLiRS could be as the chime node in a wireless doorbell system.

3.4.3 Enhanced Slave

Enhanced slaves have the same functionality as routing slaves and they are handled in the same way in the network. The difference between routing slaves and enhanced slaves is that enhanced slaves have an EEPROM for storing application data.

An example of an enhanced slave node could be a weather station.

3.4.4 Zensor Net Routing Slave

The Zensor Net routing slave node is basically a routing slave node configured as FLiRS with the additional functionalities:

- Zensor Net binding
- Zensor Net flooding

The Zensor Net is an alternative network to the classic Z-Wave network having its own binding method and a mechanism to flood messages to the entire Zensor Net. Notice that only ZDK 5.0x supports Zensor Net.

An example of a zensor net routing slave node could be a smoke detector.

3.5 Home ID and Node ID

The Z-Wave protocol uses a unique identifier called the Home ID to separate networks from each other. The Home ID is a 32 bit unique identifier that is pre programmed in all controller devices. All slave nodes in the network will initially have a home ID that is zero, and they will therefore need to have a home ID assigned to them by a controller in order to communicate with the network. Controllers in a network can exchange home ID’s so more than one controller can control slave nodes in a network.

Node ID’s are used to address individual nodes in a network, they are only unique within a network defined by a unique home ID. A node ID is an 8 bit value and like home ID’s they are assigned to slave nodes by a controller.

With the introduction of ZDK 4.50, the home ID may still be programmed. However, when excluding a controller it generates a new random home ID.
The generation of a random home ID not only simplifies production, but also eliminates the risk of creating node ID duplicates because a new home ID is used when creating the network.
With ZDK's before v4.50, this would typically happen when resetting a primary controller without first removing all nodes from the network.

With the introduction of ZDK 4.50, slave nodes also generate a new random home ID after each exclusion. During Network-Wide Inclusion, the home ID is used to identify the node. Once included, the node takes on the home ID of the primary controller and the slave's random home ID is never used again.

4 MAC LAYER

The Z-Wave MAC layer controls the radio frequency medium. The data stream is Manchester coded (true for 9600 bps transmission mode) and consists of a preamble, start of frame (SOF), frame data and an end of frame (EOF) symbol. The frame data is the part of the frame that is passed on to the transport layer.

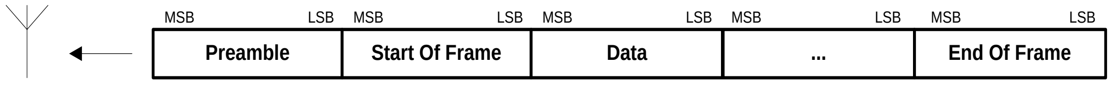


Figure 3 Z-Wave data stream

All data is sent in little endian format.

The MAC layer is independent of the RF media, frequency and modulation method but the MAC layer requires either access to the frame data when received or to the whole signal in binary form either as an decoded bit stream or to the Manchester coded bit stream.

Data are transmitted in blocks of 8bit, most significant bit first and the data is Manchester coded in order to have a DC free signal.

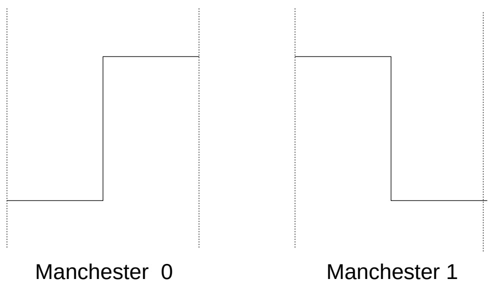


Figure 4 Manchester coding

4.1 Collision avoidance

The MAC layer has a collision avoidance mechanism that prevents nodes from starting to transmit while other nodes are transmitting. The collision avoidance is achieved by letting nodes be in receive mode when they are not transmitting, and then delay transmit if the MAC layer is currently in the data phase in the receiver. The collision avoidance is active on all types of nodes when they have the radio activated.

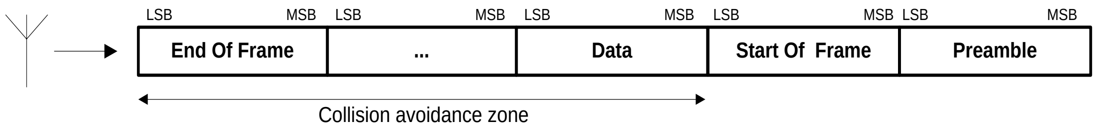


Figure 5 Collision avoidance

The transmission of the frame is postponed by a random number of milliseconds when the media is busy.

5 TRANSFER LAYER

The Z-Wave transfer layer controls the transfer of data between two nodes including retransmission, checksum check and acknowledgements.

5.1 Frame Layout

The Z-Wave transfer layer contains four basic frame formats used for transferring commands in the network. All four frames use the following frame layout:

7	6	5	4	3	2	1	0
Home ID ..							
Source Node ID							
Frame header ..							
Length							
Destination address ..							
Data byte 0-x							
..							
..							
Checksum							

Figure 6 Z-Wave basic frame format

Payload data is preceded by a header that contains information about the Z-Wave networks Home ID, source Node ID, frame header, length of frame, destination Node ID, and payload. To verify integrity of the frame, a checksum byte is appended to the end of frame.

5.1.1 Singlecast Frame Type

Singlecast frames are always transmitted to one specific node, and the frame is acknowledged so the transmitter knows that the frame has been received.

A singlecast transmission has the following frame flow.

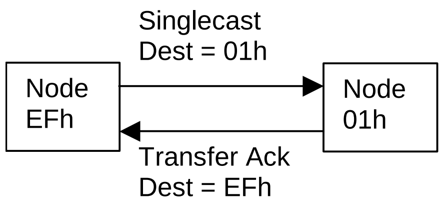


Figure 7 Singlecast transmission

If the singlecast frame or the transfer acknowledge frame is lost or corrupted, the singlecast frame is retransmitted. In order to avoid potential collisions with parallel systems the retransmissions are delayed with a random delay. The random delay must be in steps of the time it takes to send a frame of the maximum frame size and receive the Transfer Ack.

The singlecast frame can optionally be used without acknowledgement in a system where reliable communication isn't required.

5.1.2 Transfer Acknowledge Frame Type

The transfer acknowledge is a Z-Wave singlecast frame where the size of the data section is zero. For further description of the singlecast frame see section 0

5.1.3 Multicast Frame Type

Multicast frames are transmitted to a number of nodes ranging from 1 to 232 nodes. This frame type doesn't support acknowledge.

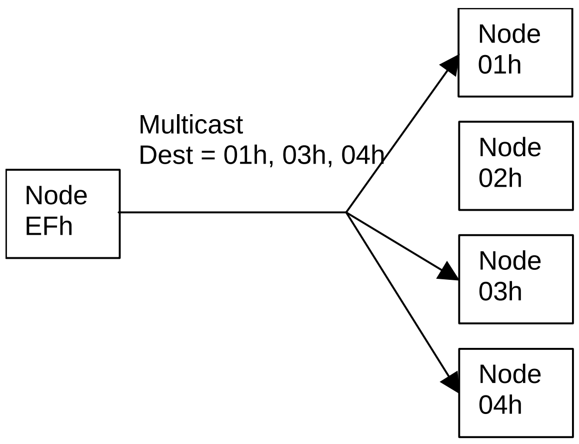


Figure 8 Multicast transmission

The multicast destination address is used to address selected nodes without having to send a separate frame to each node.

Note that a multicast frame doesn't get acknowledged so this type of frame can't be used for reliable communication. If reliable communication is needed a multicast must be followed by a singlecast frame to each destination node.

5.1.4 Broadcast Frame Type

Broadcast frames are received by all nodes in a network, and the frame is not acknowledged by any nodes.

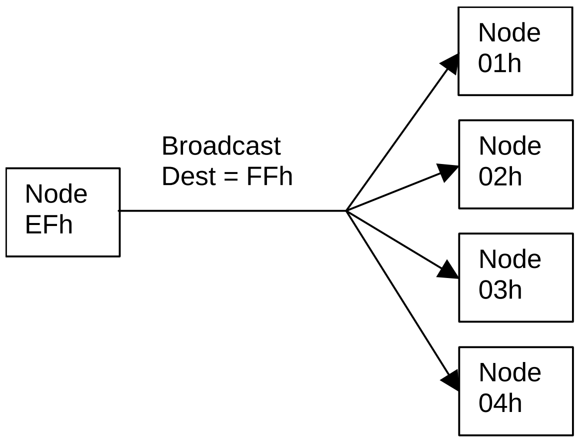


Figure 9 Broadcast transmission

Note that a broadcast frame doesn't get acknowledged so this type of frame can't be used for reliable transfer. If reliable communication is needed a broadcast must be followed by a singlecast frame to each destination node.

5.1.4.1 Explorer Frame Type

Explorer frames are a special class of broadcast frames. Being broadcasted, all nodes in direct range of the originator receive an explorer frame.
If supporting explorer frames, the protocol layer interprets the frame to determine a number of properties:

- **Addressing**
    - Is the explorer frame header addressing this particular node?
    - Is the explorer frame header addressing all explorer nodes?
- **Forwarding**
    - Is the frame to be forwarded by this node?
    - Is the frame to be forwarded by all nodes?
- **Congestion control**
    - Should the node discard all frames resembling this frame?
- **Inclusion management**
    - Is the explorer frame header carrying an InclusionRequest?

The actual explorer frame properties depend on the purpose of the specific explorer frames.

A SearchRequest addresses a particular node but must be forwarded by all nodes.
A SearchResult addresses a particular node and must be forwarded only by all nodes in a route.
A SearchStop addresses a particular node and must be forwarded only by all nodes in a route but all nodes must discard frames resembling the frame (identified by srcNodeId + seqNo).

6 ROUTING LAYER

The Z-Wave routing layer controls the routing of frames from one node to another. Both controllers and slaves can participate in routing of frames in case they are always listening and have a static position. The layer is responsible for both sending a frame with a correct repeater list, and also to ensure that the frame is repeated from node to node. The routing layer is also responsible for scanning the network topology and maintaining a routing table in the controller.

6.1 Frame Layout

The Z-Wave routing layer has 2 kinds of frames that are used when routing of frames is necessary.

6.1.1 Routed Singlecast Frame Type

The Z-Wave routed singlecast is a one-node destination frame with acknowledge that contains router information. The frame is repeated from one router to another until it reaches its destination.

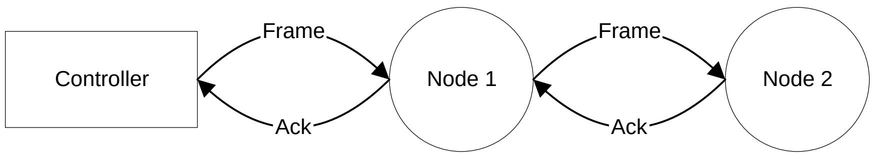


Figure 10 Routed singlecast transmission

6.1.2 Routed Acknowledge Frame Type

The Z-Wave routed acknowledge is a routed singlecast frame without payload that is used to tell the controller that the routed singlecast has reached its destination.

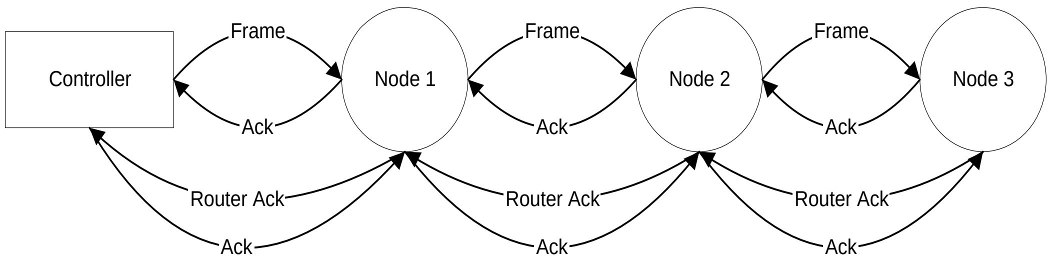
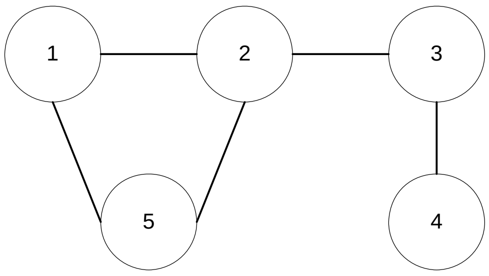


Figure 11 Routed singlecast transmission

ZDK 4.2x introduced silent acknowledge reducing the necessary frames by close to 50% and thereby decreasing latency.

6.2 Routing Table

The routing table is where a controller keeps the network topology. The table is a bit field table storing all information about what nodes that can see each other. The figure below illustrates a network topology and the resulting routing table.



	1	2	3	4	5	6
1	0	1	0	0	1	0
2	1	0	1	0	1	0
3	0	1	0	1	0	0
4	0	0	1	0	0	0
5	1	1	0	0	0	0
6	0	0	0	0	0	0

Figure 12 Network topology and routing table

The routing table is build by the primary controller based on information it receives from all the nodes in the network, at installation time, about each nodes range.

6.3 Route to Node

Finding the route to a node is a difficult task because a portable controller is defined as a device that will be moved around a lot (e.g. a remote control). Therefore, a portable controller will always try to reach a node without routing and if that fails the portable controller will use several techniques to find the best route to the node.

6.4 Explorer Route Resolution

With the introduction of ZDK 4.50, nodes have a new, flexible method for locating other nodes, called explorer route resolution.

Explorer route resolution is a supplement to the existing routing table mechanism.
Explorer route resolution is not backwards compatible with versions earlier than ZDK 4.50.
The protocol layer maintains full support for classic routing as it works in earlier versions.

Nodes supporting explorer route resolution have significantly better chances of re-locating a lost target within a short time. A target not supporting explorer route resolution must be located via classic routing methods. First, all possible routes of the routing table may be tried. If that also fails, a complete network re-discovery may be performed.

6.4.1 Last Working Route

Slightly re-factored, the Last Working Route is a re-use of the buffer storage known as the ResponseRoute. When an application issues a SendData API call, the protocol layer tries the LastWorkingRoute as a first approach.

Typically, the LastWorkingRoute holds a valid route to the target, thus providing the fastest access to the target and saving the network from unnecessary discovery traffic.

If attempts to communicate via the LastWorkingRoute fail, the protocol layers may resort to the explorer SearchRequest.

It is standard behavior for a portable controller to try direct range before trying anything else, so chances are that the target is not within direct range.
A static controller does not start by trying direct range. There is however, no need to try another direct range communication.

6.4.2 The Explorer Search process

SearchRequest and SearchStop frames may typically follow an explorer SearchRequest reaching its target within direct range so quickly that no other explorer nodes forward copies of the SearchRequest before all copies have been discarded again.

In case the target is not located within direct range, all nodes that received the SearchRequest forwards a copy of the frame. The forwarding algorithm uses random scheduling to reduce the risk of collisions.
In addition, the algorithm discards all additional copies and at same the time ensures that no frames form loops.

An explorer frame may be forwarded over up to 4 hops. A TTL (Time-To-Live) mechanism ensures that frames cannot travel longer paths around the network.

7 APPLICATION LAYER

The Z-Wave application layer is responsible for decoding and executing commands in a Z-Wave network. The only part of the application layer that is described in this overview is the assignment of Home ID's and Node ID's and the replication of controllers. The rest of the application layer is implementation specific, and can be different from one implementation to another.

7.1 Frame Layout

The frame format used in the Z-Wave application layer is described in this section.

7.1.1 Application Layer Frame Format

7	6	5	4	3	2	1	0
Single/Multi/Broadcast frame header							
Application command class							
Application command							
Command parameter 1							
Command parameter 2							
...							
Command parameter x							

Figure 13 Z-Wave application frame format

Application command class:

The application command class specifies which class of commands the command belongs to.

Currently defined command classes:

Command Class	Description
00h-1Fh	Reserved for the Z-Wave Protocol
20h-FFh	Reserved for the Z-Wave Application

Figure 14 Z-Wave command class range

Application command:

The application command specifies the specific command or action within the command class. The commands in the Application specific command classes are described in [1] and [2].

Command parameter 1-x:

The command parameters contain any parameters associated with the specified command. The number of parameters depends on the command.

All frame types except acknowledge can contain an application command.

7.2 Node information

Because a controller in a Z-Wave network should be able to control many different kinds of nodes, it is necessary to have a frame that describes the capabilities of a node. Some of the capabilities will be protocol related and some will be application specific. All nodes will automatically send out their node information when the action button on the node is pressed. A controller can also get the node information from a node by requesting it with a “get node information” frame.

7.2.1 Node Information Frame Flow

The node information frame is send out by a node each time its action button is pressed. The frame is sent out as a broadcast to any controller/node that might be interested in the information. A controller can also request the node information from a node by sending a get node information frame to it.

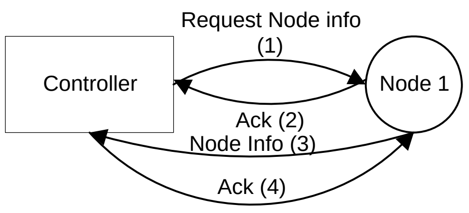


Figure 15 Get node info frame flow

8 REFERENCES

[1] Zensys, SDS10242, Software Design Specification, Z-Wave Device Class Specification
[2] Zensys, SDS11060, Software Design Specification, Z-Wave Command Class Specification