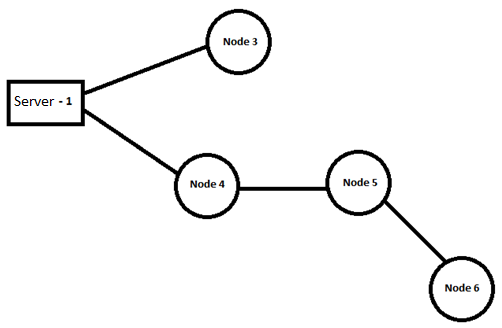
# Communication protocol



As stated in the system overview, when a node is too far away from the **Server** to get the signal from the **Server**, so we make a communication protocol between node to send and received information.

Our protocol uses Tree topology. The central 'root' node is called Server. Server connects to other nodes with a point-to-point link. The point-to-point link is RF12 library.

Sending information between nodes and server bases on “intermediate node” called repeater. This repeater will be responsible for capturing data from the **Server** and deliver this data to the far-away node or vice versa. To be general in all cases, we will define a data frame to send back and forth among nodes.

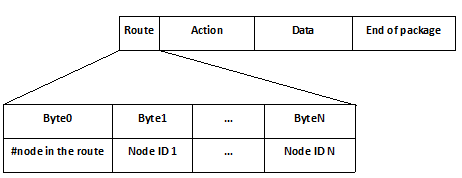
## 1. Frame

When a node wants to send information to other nodes, it has to send the information compressed in the following frame:

|  |  |  |  |
| --- | --- | --- | --- |
| **Route** | **Action** | **Data** | **End of package** |

### 1.1 Route

The **Route** of the frame will define the path of the transmission from the sender to the target. It has the following format:



**Byte0:** indicates how many nodes there are in the path (exclusive the sender).

**Byte1 – ByteN:** Indicates **NodeIDs** the information must go through from the sender to the destination node.

**Node ID:** the identification number of a node.

### 1.2 Action

**Action** part contains **one** byte with one of the following characters:

|  |  |
| --- | --- |
| ***Value*** | ***Type of Action*** |
| 77 (ASCII number of ‘M’) | The **Server** sends the **map** to the destination node so that the destination node can know the way to send data back to the central. This is used only when **Server** sends data to other nodes. When the destination node gets the map, it will send the acknowledgement back to Server. |
| 82 (ASCII number of ‘R’) | The sender **requests** data from the destination node. |
| 68 (ASCII number of ‘D’) | The sender sends **data** to the destination node. |
| 65 (ASCII number of ‘A’) | The sender sends **acknowledgement** to the destination node. |
| 66 (ASCII number of ‘B’) | The sender sends report bad action when it doesn’t understand the Action. |
| 83 (ASCII number of ‘S’) | The sender sends **setting parameter** to the destination node. When the destination node gets the setting parameter, it will send the acknowledgement back to **Server**. |
| 88 (ASCII number of ‘X’) | The sender send the control information to X10 system |
| 74 (ASCII number of ‘J’) | The sender send the control information to Motor |

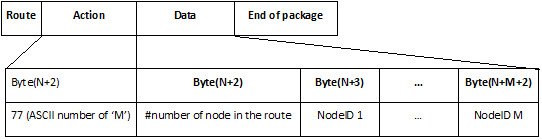
|  |
| --- |
| Byte(N+1) |
| Type of Action |

### 1.3 Data

**Data** depends on the action the sender has sent in the previous byte.

1. Action is **‘M’**: the first byte of **Data** will be the number of nodes in the path from the destination back to the **Server**. The next bytes will be IDs of these nodes which are on the routes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte(N+2)** | **Byte(N+3)** | **…** | **Byte(N+M+2)** |
| #number of node in the route | NodeID 1 | … | NodeID M |



1. Action is **‘R’**: **Data** will contain **one byte** standing for names of sensors that sender is requesting. This byte will be one of the following characters:

|  |  |
| --- | --- |
| ***Type of Request*** | ***Value*** |
| Humidity, Temperature, Light. | 65 (ASCII number of ‘A’) |
| Water level | 87 (ASCII number of ‘W’) |

|  |
| --- |
| Byte(N+2) |
| Type of Request |

1. Action is ‘**D**’: **Data** contains **bytes** which are information from sensors. It has the following format:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte1** | **Byte2** | **Byte3** | **…** | **ByteM** | **Byte(M+1)** | **Byte(M+2)** | **…** | **Byte(M+k)** | **…** |
| # of sensors | Type of 1st sensors | # of data bytes | | | Type of 2st sensors | Number of data bytes | | |  |

|  |  |  |
| --- | --- | --- |
| ***Type of Sensor*** | ***Value of Byte2*** | ***Byte3…ByteM*** |
| Humidity | 72 (ASCII number of ‘H’) | Next 4 bytes is an float variable store the raw data of humidity sensor |
| Light | 76 (ASCII number of ‘L’) | Next 4 bytes is an integer variable store the raw data of light sensor |
| Temperature | 84 (ASCII number of ‘T’) | Next 4 bytes is an float variable store the raw data of temperature sensor |
| Occupancy | 79 (ASCII number of ‘O’) | Next 1 byte is a byte variable store the data of Occupancy sensor (‘Y’/’N’ or 89/78) |
| Water level | 87 (ASCII number of ‘W’) | Next 4 bytes is an float variable store the raw data of ultrasonic sensor (distance in centimeter) |
| Power | 80 (ASCII number of ‘P’) | Next 4 bytes is an float variable store the raw data of power sensor. |
| Water flow | 71 (ASCII number of ‘G’) | Next 4 bytes is an integer variable store the raw data of water flow sensor. |

1. Action is ‘**A**’: **Data** will contain **one byte** standing for type of acknowledgment:

|  |  |
| --- | --- |
| ***Type of acknowledgment*** | ***Value*** |
| Received map | 77 (ASCII number of ‘M’) |
| Received setting parameter | 83 (ASCII number of ‘S’) |
| Received Occupancy data | 79 (ASCII number of ‘O’) |
| Received Power data | 80 (ASCII number of ‘P’) |
| Received Water flow data | 71 (ASCII number of ‘G’) |
| Received X10 data | 88 (ASCII number of ‘X’) |
| Received Motor data | 74 (ASCII number of ‘J’) |

|  |
| --- |
| Byte(N+2) |
| Type of acknowledgment |

1. Action is ‘**S**’: **Data** will contain **one byte** is the value of setting parameter:

|  |
| --- |
| Byte(N+2) |
| Value of setting parameter |

1. Action is **‘J’**: **Data** will contain **one byte** is the value of controlling information of motor:

|  |
| --- |
| Byte(N+2) |
| Value of Control information |

|  |  |
| --- | --- |
| ***Value of Control information*** | ***Control Action*** |
| 0x00 | Turn off the motor |
| 0x02 | Turn motor right |
| 0x03 | Turn motor left |

1. Action is **‘X’**: **Data** will contain **8 byte** is the value of controlling parameter for X10:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte(N+2) | Byte(N+3) | Byte(N+4) | Byte(N+5) | Byte(N+6) | Byte(N+7) | Byte(N+8) | Byte(N+9) |
| H | house | U | unit | C | command | D | dim |

|  |  |
| --- | --- |
| ***Value of Control information*** | ***Control Action*** |
| house | House name of X10 device |
| unit | Unit name of X10 device |
| command | Command of X10 |
| dim | Use the dim light |

### 1.4 End of package

End of package of **Node** is 3 bytes:

|  |  |  |
| --- | --- | --- |
| Byte(L-2) | Byte(L-1) | ByteL |
| 73 (ASCII number of ‘I’) | Node ID | 255 |

**Node ID** is the identification number of current node.

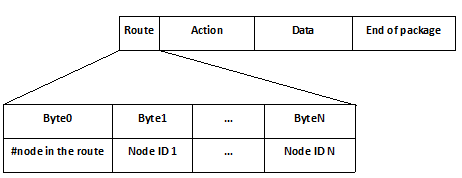
End of package of **Server** is 1 bytes:

|  |
| --- |
| ByteL |
| 255 |

## 2. Behavior

### 2.1 Repeater

When a node receives information from another node, the data packet will follow the frame:

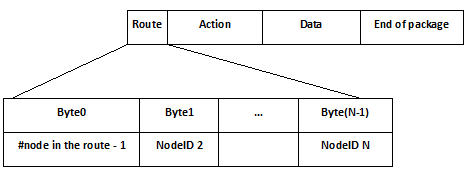


The node will check the value of byte0 (the number of node in the route). If the byte0 is equal 1, the node is the destination.

If byte0 is larger than 1, the node is repeater, it will change the information of route, follow the instruction:

* Decrease the number of node in the route by 1. (byte0 =byte0 -1)
* Remove its NodeID out of route. (remove byte1)

The new **Route** will be:



The new data packet will send to NodeID 2.

### 2.2 Acknowledgements

When the node receives the map information, it will send the acknowledgement back to server. (Receive action ‘M’, respond by action ‘A’).

When the Server receives the Occupancy data, it will send the acknowledgement back to node. (Receive action ‘D’, respond by action ‘A’).

Another acknowledgement:

|  |  |  |
| --- | --- | --- |
| ***Type of acknowledgment*** | ***Value*** | ***Direction*** |
| Received map | 77 (ASCII number of ‘M’) | Node -> Server |
| Received setting parameter | 83 (ASCII number of ‘S’) | Node -> Server |
| Received Occupancy data | 79 (ASCII number of ‘O’) | Server -> Node |
| Received Power data | 80 (ASCII number of ‘P’) | Server -> Node |
| Received Water flow data | 71 (ASCII number of ‘G’) | Server -> Node |
| Received X10 data | 88 (ASCII number of ‘X’) | Node -> Server |
| Received Motor data | 74 (ASCII number of ‘J’) | Node -> Server |

### 2.3 Request information to sensor

When the node receives the request information, it will send the data of this request to server.

1. Humidity, Temperature, Light sensor
2. Water level sensor.

(Receive action ‘R’, respond by action ‘D’).

### 2.4 Auto send sensor data information

Base on some conditions, the node will send information to the Server. The server will send the Acknowledgment back to the node. If the node doesn’t receive the acknowledgment, it will resend the data.

1. Occupancy

2. Water flow

3. Power monitor

### 2.5 Send control information

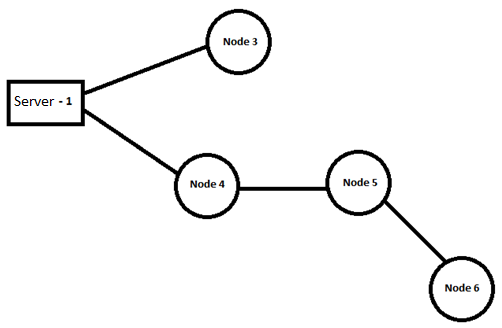
Base on some conditions, the Server will send control information to the node. The node will send the Acknowledgment back to the Server. If the Server doesn’t receive the acknowledgment, it will resend the data.

1. X10 System

2. Motor

## 3. Example

The example will apply the protocol as define above in below map.



The ID number of Server node is 1.

The ID number of Node 3, Node 4, Node 5, and Node 6 are 3, 4, 5 and 6.

### 3.1 Maps

#### 3.1.1 Send maps to node 3

1. The Server sends bytes to Node 3:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 77 | 1 | 1 | 255 |

2. Node 3 send the bytes to Server:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 65 | 77 | 73 | 3 | 255 |

#### 3.1.2 Send maps to node 6

Node 4 and node 5 will be repeaters.

Server sends maps to Node 6 following steps:

1. The Server sends bytes to Node 4:

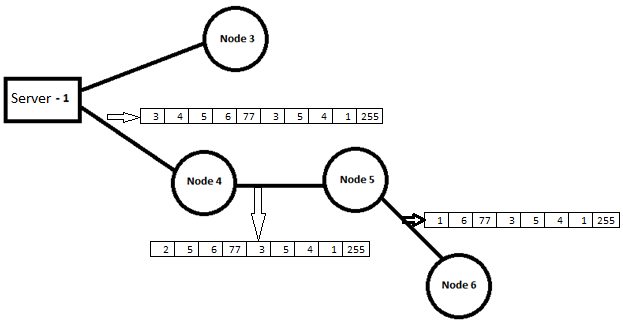
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 5 | 6 | 77 | 3 | 5 | 4 | 1 | 255 |

2. The Node 4 sends bytes to Node 5:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 6 | 77 | 3 | 5 | 4 | 1 | 255 |

3. The Node 5 sends bytes to Node 6:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 6 | 77 | 3 | 5 | 4 | 1 | 255 |



4. When Node 6 receives the map, it will send back the map acknowledgment to Server:

Node 6 🡪 Node 5:

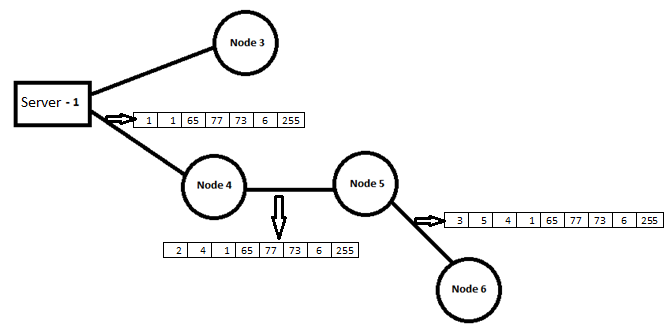
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 5 | 4 | 1 | 65 | 77 | 73 | 6 | 255 |

Node 5 🡪 Node 4:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 4 | 1 | 65 | 77 | 73 | 6 | 255 |

Node 4 🡪 Server:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 65 | 77 | 73 | 6 | 255 |



### 3.2 Request the information about humidity, temperature and light sensor from node 6

1. The Server sends request to node 6:

Server 🡪 Node 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 5 | 6 | 82 | 65 | 255 |

Node 4 🡪 Node 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | 5 | 6 | 82 | 65 | 255 |

Node 5 -> Node 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 6 | 82 | 65 | 255 |

2. The Node 6 sends information to Server:

Temperature: 00C. Humidity: 0 %. Light: 0

Node 6 🡪 Node 5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 5 | 4 | 1 | 68 | 3 | 72 | 0 | 0 | 0 | 0 | 84 | 0 | 0 | 0 | 0 | 76 | 0 | 0 | 0 | 0 | 73 | 6 | 255 |

Node 5 🡪 Node 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 4 | 1 | 68 | 3 | 72 | 0 | 0 | 0 | 0 | 84 | 0 | 0 | 0 | 0 | 76 | 0 | 0 | 0 | 0 | 73 | 6 | 255 |

Node 4 🡪 Server

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 68 | 3 | 72 | 0 | 0 | 0 | 0 | 84 | 0 | 0 | 0 | 0 | 76 | 0 | 0 | 0 | 0 | 73 | 6 | 255 |

### 3.3 Auto send information about Occupancy sensor from node 6

1. The Node 6 sends information to Server:

Occupancy information: ‘Y’ (#89)

Node 6 🡪 Node 5

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 5 | 4 | 1 | 68 | 1 | 79 | 89 | 73 | 6 | 255 |

Node 5 🡪 Node 4

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 4 | 1 | 68 | 1 | 79 | 89 | 73 | 6 | 255 |

Node 4 🡪 Server

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 68 | 1 | 79 | 89 | 73 | 6 | 255 |

2. The Server Acknowledgment to node 6:

Server 🡪 Node 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 5 | 6 | 65 | 76 | 255 |

Node 4 🡪 Node 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | 5 | 6 | 65 | 76 | 255 |

Node 5 -> Node 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 6 | 65 | 76 | 255 |

In summary, with this communication protocol, we are able to use just one program but applicable for all nodes in the wireless network. It does not matter the nodes are near the Servers (no repeater required) or very far away from the Server (many repeater required). Regularity and modularity are the key advantages of this protocol.