# Analysis of Software Defined WSN and Related Network Re-orchestration Down-Time

Indrajit S. Acharyya(✉) and Adnan Al-Anbuky

Sensor Network and Smart Environment Research Centre, Auckland University of Technology, Auckland, New Zealand
iarchary@aut.ac.nz

**Abstract.** Re-orchestration is a crucial operational requirement secured by Software-Defined Wireless Sensor Networks (SDWSN). Herein, WSN flexibility involves functional identification, definition, modularization, and virtualization. These components support sensor network software-control and allow for dynamically redefining the network topology and operational behavior. This paper reflects the core ideology behind the proposed cloud-based cyber-physical organizational structure. This utilizes the virtualization and testing of the functionalities that edge towards desired re-orchestrations. The physical implementation of re-orchestration process, however, may cause network operational disruption for a brief period of time. This research work reflects the parameters involved in network re-orchestration process. Analysis of an example performance indicator that influences the downtime in SDWSN during re-orchestration is offered. Clarity of these involvements can offer significant help in supporting planning for reducing this 'down time'.

**Keyword:** SDWSN · Network virtualization · Re-orchestration latency

## 1 Introduction

Re-orchestration of wireless sensor network organizations tends is a key requirement for reflexive adaptation towards dynamic service demands [1–6]. A system solution involving operation of a 'Software-defined wireless sensor network' (SDWSN) under the aegis of a Cyber-Physical System (CPS) organization serves as a viable approach towards offering support for such operational flexibility in a dynamic manner [1–4, 7–15]. One of the approaches for software redefinition is that of 'Reconfigurability'. Here, the physical devices could be re-configured to certain roles or possibly even assume multiple functional roles simultaneously [3, 4]. The presence of requisite 'logical software modules' residing within the code with which they were initially configured allows for assuming multiple configurable status. This approach requires the 'program' or 'code' (with which the devices are configured) to comprise of requisite conditional statements as well as 'well-defined', 'logical' modules that could be assumed (in accordance with conditional execution within the program) during run-time. Prior to embarking upon establishment of such a 'modular' software defined sensor network, it is deemed imperative to reflect certain essential pre-requisites that could arguably be entailed by such

https://doi.org/10.1007/978-3-030-89170-1_6

organizations from a formalization standpoint. These pre-requisites pertain to identification and definition of the integral functional components associated with any sensor network formation, modularization of the functions so identified (based on their definitions) and subsequently paving the way for their virtualization [3, 4].

It is deemed conceivable to assert that any sensor network is composed of the three generic functions viz., ‘Gateway’, ‘Leaf’ and ‘Routing’ [3, 4]. By means of including certain examples of possible topological orientations that a sensor network could resort to (by virtue of SDWSN re-orchestration), a specific portion of the preceding work [4] brings to the fore the integrity of the elementary functionalities in regard to any sensor network organization. The motive behind ensuing upon modularizing and virtualizing of the three key functionalities (so identified and defined) is to render them as re-usable virtual modules that could be subjected to soft-trials within the cloud-hosted virtualization environment. Such virtualization-abetted ‘re-orchestration planning’ could lend itself towards significant expedition of the re-orchestration implementation process. The scope of the re-orchestration implementation process, however, may yet entail a certain amount of latency that could prove to be detrimental to its ongoing processes [3, 4].

The research work documented within the previous work [4] highlighted the ideological basis (along the lines of Industry 4.0) leading to the proposed modular architectural organization for software-defined sensor network. Furthermore (by means of certain example cases of network re-orchestration), it also highlighted the efficacy of a virtual environment towards ‘planning’ the desired re-orchestrations as well as gaining an insight into the key performance measure of the ‘downtime’ or latency experienced by the network whilst undergoing re-orchestration. In furtherance to the research work documented within [4], this paper vies to put forth the pseudo codes for each of the core WSN functions, outlining the operational activities associated with each, and thereby attempting to explore how ‘software manipulation’ of certain of these key network parameters viz., MAC protocol employed, data communication rate, etc. could influence network ‘re-orchestration downtime’. Besides, certain research work pertaining to network downtime has also been included as part of the literature review herein.

In furtherance to the state-of-the-art pertaining to SDN-facilitated topological re-orchestration of WSNs documented within [4], literature review revolving around the time elapsed as a result of a network undergoing re-orchestration has been considered within this paper. Zhou et al. [16] seek to undertake an approach directed towards effectively eliminating any re-orchestration-induced downtime via partitioning the network into multiple subsets and allowing for each of them to undergo re-orchestration sequentially i.e. one subset at a time. For this purpose, the authors ensue upon formulating the problem of partitioning the sensor network in subsets and proving the same to be ‘Non-deterministic polynomial time. Subsequently, heuristics pertaining to downtime free system migration are presented and compared.

Szczodrak et al. [17] propose a ‘Fennec Fox’, a framework that allows for flexible WSN reconfiguration via offering supporting for different applications that could be dynamically switched to, at different time instants (based on the network service requirements). Each such application is based on separate MAC and network protocol specifications. Network re-orchestrations are brought about by means of broadcasting of ‘control messages’ amongst the (relevant) nodes in a sort of peer-to-peer fashion. The authors utilize the ‘Finite State Model’ computation model in a bid to model such re-orchestrations undergone by the network. Based on experimentation conducted, authors mention that adoption of such an approach results in a reconfiguration delay of the order of milliseconds and that it (reconfiguration delay) is dependent on factors such as distance between the nodes, radio-duty cycling, etc. and that as the reconfiguration delay decreases, so does the percentage of the number of nodes that successfully get configured.

## 2 Main WSN Functional Modules

As alluded to within the earlier section, the three key functions that enable a WSN to perform its tasks satisfactorily pertain to sensing and data acquisition (Sensing function), routing (Router function) and escalation of data aggregated by a sink node (Gateway function) [3, 4]. The core and auxiliary activities associated with each of these three core functions is depicted in Fig. 1 [3, 4].

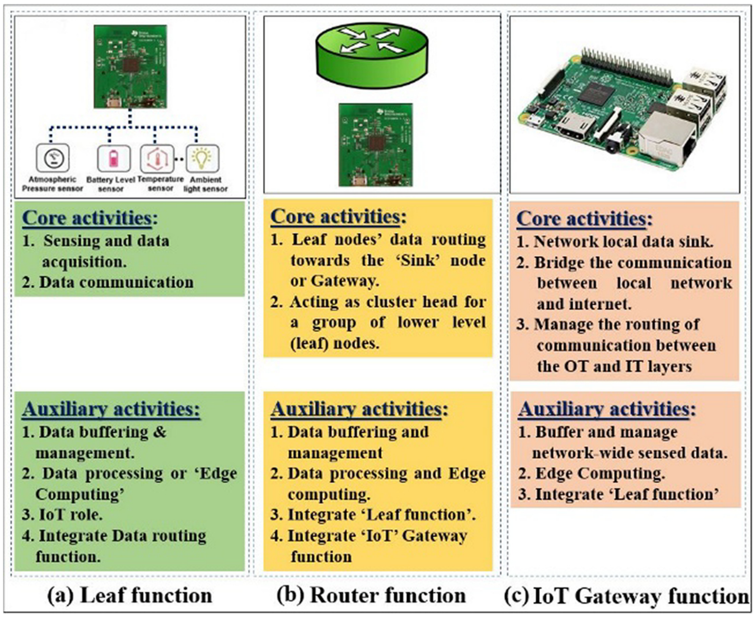

Atmospheric Pressure sensor
Battery Level sensor
Temperature sensor
Ambient light sensor
Core activities:
1. Sensing and data acquisition.
2. Data communication
Auxiliary activities:
1. Data buffering & management.
2. Data processing or ‘Edge Computing’
3. IoT role.
4. Integrate Data routing function.
(a) Leaf function
Core activities:
1. Leaf nodes’ data routing towards the ‘Sink’ node or Gateway.
2. Acting as cluster head for a group of lower level (leaf) nodes.
Auxiliary activities:
1. Data buffering and management
2. Data processing and Edge computing.
3. Integrate ‘Leaf function’.
4. Integrate ‘IoT’ Gateway function
(b) Router function
Core activities:
1. Network local data sink.
2. Bridge the communication between local network and internet.
3. Manage the routing of communication between the OT and IT layers
Auxiliary activities:
1. Buffer and manage network-wide sensed data.
2. Edge Computing.
3. Integrate ‘Leaf function’
(c) IoT Gateway function

**Fig. 1.** Core and non-core activities associated with the main WSN functional modules (a) Leaf node function (b) Router node function and (c) IoT-based WSN Gateway node function [3, 4].

Such clear segregation of the main WSN functions on the basis of the unique core activities associated with each of them paves the way for rendering them as reusable modules [3, 4]. Such independent functional modules could be assumed one at a time or simultaneously (provided the hardware employed is able to encompass and execute multiple tasks at the same time. Modern day SoCs-based wireless sensor transceivers are capable of accommodating for edge-computing-based tasks.). These (definition and modularization) pave the way for enhanced flexibility from the node-operational standpoint. Such node-level flexibility may implicate on the flow of data within the network, in some cases altering the topology of a given network (network-level flexibility). In this regard, consider Fig. 2 wherein certain different possible (typical) network topological arrangements are presented [4]. Herein, the CC538 Evaluation Module (EM), along with the Raspberry Pi that have been utilized for implementation purposes within this research have been depicted. Most of the CC2538 EMs have been configured to play the role of leaf nodes whereas a few have been configured to execute the role of routers. The gateway function, however, has been realized by means of employing the Raspberry Pi which acts as protocol converter and escalates the sensed data over to the cloud [3, 4].

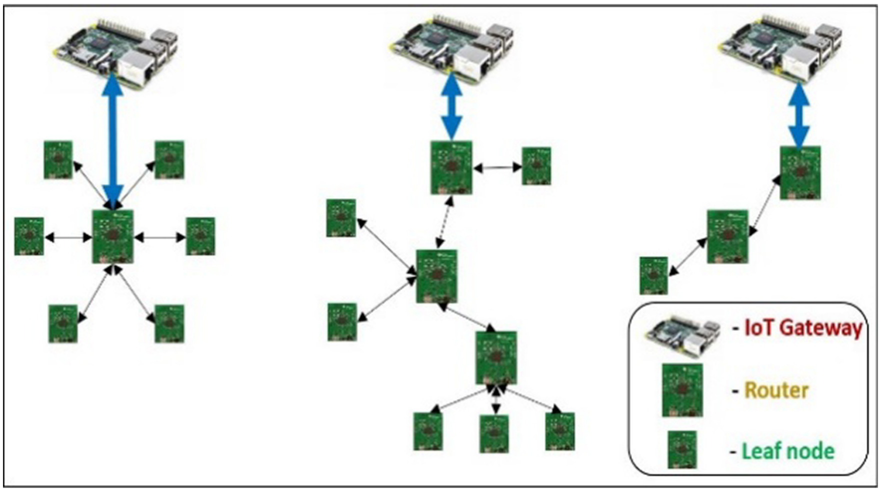


**Fig. 2.** Typical WSN Topological arrangements [3, 4].

Contiki software (IDE) has been employed for developing and compiling of codes as well as configuration of the physical hardware nodes (using the codes so developed and compiled). By virtue of its network simulator platform of Cooja, Contiki also provisions for network virtualization (since the same codes utilized for hardware nodes can be used to create and/or configure virtual nodes), typically within the cloud server. Besides monitoring of the underlying physical WSN, such a virtualization platform can be availed for soft trialing of numerous re-orchestration scenarios, prior to implementation on the physical network.

## 3 SDWSN Proposed Architecture

Based on the ideology prescribed by the Industry 4.0 paradigm, the architectural proposition herein consists of two layers viz., ‘Operational Technology (i.e. OT)’ layer and ‘Information Technology (i.e. IT)’ layers, as depicted in Fig. 3. The upper layer of IT hosts the three main components of the ‘Virtualization unit’, ‘Data and Knowledge

Repository' and the requisite 'Operational software' that act as facilitators for trialing and figuring out the necessary re-orchestrations to be applied to the lower layer of OT which hosts the physical hardware nodes [3, 4].

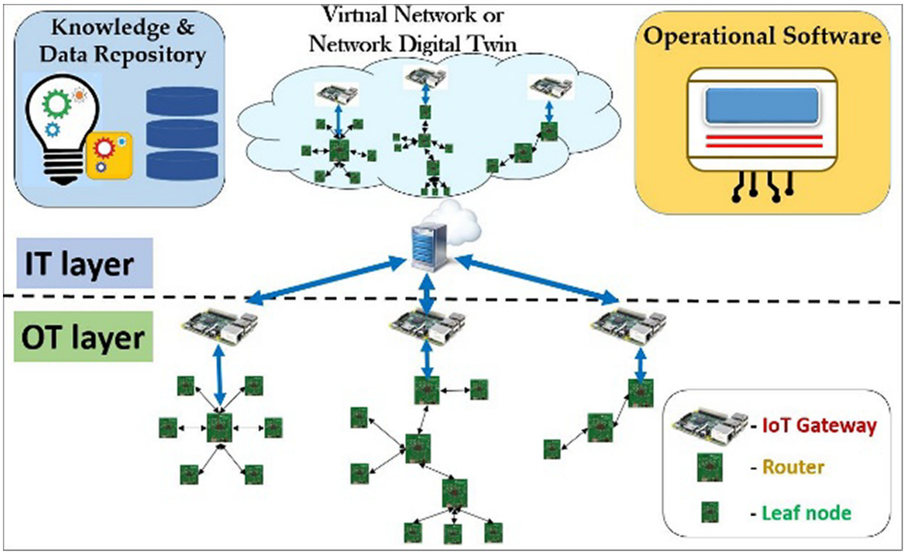


**Fig. 3.** Architectural proposition for software-defined sensor network based on the ideology of the Industry 4.0 paradigm [3, 4].

### 3.1 IT Layer

Besides monitoring, the component of virtualization unit facilitates plays a key role towards planning of the suitable re-orchestration scenarios that could be assumed by the underlying physical layer. Such suitable re-orchestrations are attained through running soft trial of re-orchestration scenarios and observing the implications on the network performance for the same. Besides cutting costs, such hardware-independent testing reduces the time required for working out the 'customized' or 'most pertinent' re-orchestrations, allowing for the system to better cope with the real-time re-orchestration service demands [3, 4].

The virtualization unit however, requires interaction with both the OT layer (to fetch the necessary parametric information required for the 'planning' of the 're-orchestrations) as well as with the 'Data and Knowledge Repository' present within the IT layer.

In a bid to assist the process of figuring out of the most suitable re-orchestrations via both augmenting the options of flexibilities available for incorporation as well as storing of re-orchestration solution applied for previous service demands, it was deemed reasonable to establish a single repository consisting of reusable virtual software modules and capable of evolving with historical experiences. Such a repository that could be readily accessed by the virtualization unit whenever required could lend itself towards tackling re-orchestration demands in a time-efficient manner [3, 4].

The ‘operational software’ unit refers to the requisite tools that allow for the development of the necessary codes pertaining to the network functionalities, as well as the necessary re-orchestration service specific knowledge components. Contiki IDE, for instance has been employed for the purposes of development, compilation, and configuration of codes for both virtual and physical nodes. Besides, software tools as MATLAB that allow for data analytics, data processing, data computation, etc. could also prove to be of vital assistance during the ‘Re-orchestration planning’ phase [3, 4].

### 3.2 OT Layer

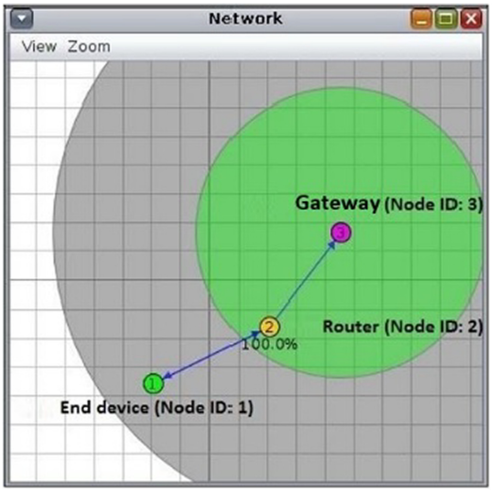
The ‘Operational Technology’ (i.e. OT) layer comprises of the physical wireless nodes deployed across the monitored area for capturing real-world data and escalate it to IT layer over the internet [3, 4]. The wireless network formed by these sensor nodes could either be clustered or non-clustered based on the application or ‘service requisites. Furthermore, each cluster (or the non-clustered physical WSN as a whole) could be centralized or decentralized. Centralized constituent clusters could be arranged in accordance with star, tree, or mesh topological frameworks (depending upon the application and/or operational requirements) whereas decentralized networks would be arranged in accordance with the mesh topological framework. Depending upon their capability and resourcefulness, codes for (certain advanced) modern day wireless transceivers (such as the Texas Instruments CC2538 SoC) could be written in such a way that they could dynamically configured to switch form their existing role of say, a leaf node to that of a router node (or vice-versa) via directing them to do so by means of an external command (as part of a re-orchestration process). Such modern SoCs could also be similarly re-configured to execute edge-computing based tasks.

## 4 Example WSN Re-orchestration Scenarios

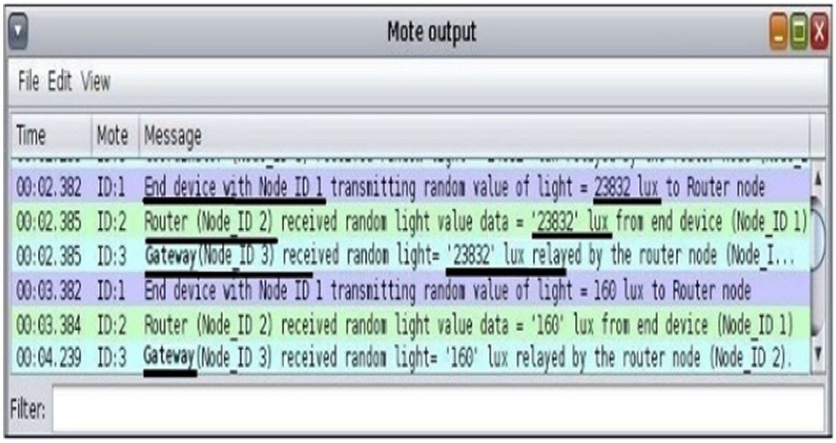
### 4.1 WSN Topological Manipulation

An example case of network topological manipulation involving reconfiguring the individual node’s functional level so to alter overall network behavior in terms of flow of data within the network [4] has been depicted within this sub-section. Consider Fig. 4a wherein a virtual three-node multi-hop network has been implemented within Cooja. Herein, the data transmitted by the node configured to act as the leaf node. In this case, node 1 reaches node 3 (the Gateway node) via the intermediate node 2 (the router node). The ‘network window’ and ‘mote output’ screenshots for the same are depicted within Figs. 5a and 5b respectively.

By means of directing (the intermediate) node acting as a router i.e. node 2 to assume leaf node function and by manipulation of the MAC protocol so as to enable the network to operate as per TDMA scheme, the same three-node network undergoes re-orchestration from the network i.e. topological standpoint and operates as a star topology based network. The ‘network window’ and ‘mote output’ screenshots for this star



**(a)**



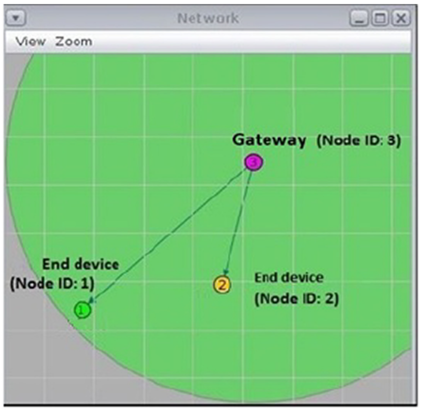
```
00:02.382 ID:1 End device with Node ID 1 transmitting random value of light = 23832 lux to Router node
00:02.385 ID:2 Router (Node_ID 2) received random light value data = '23832' lux from end device (Node_ID 1)
00:02.385 ID:3 Gateway(Node_ID 3) received random light= '23832' lux relayed by the router node (Node_I...
00:03.382 ID:1 End device with Node ID 1 transmitting random value of light = 160 lux to Router node
00:03.384 ID:2 Router (Node_ID 2) received random light value data = '160' lux from end device (Node_ID 1)
00:04.239 ID:3 Gateway(Node_ID 3) received random light= '160' lux relayed by the router node (Node_ID 2).
```

**(b)**

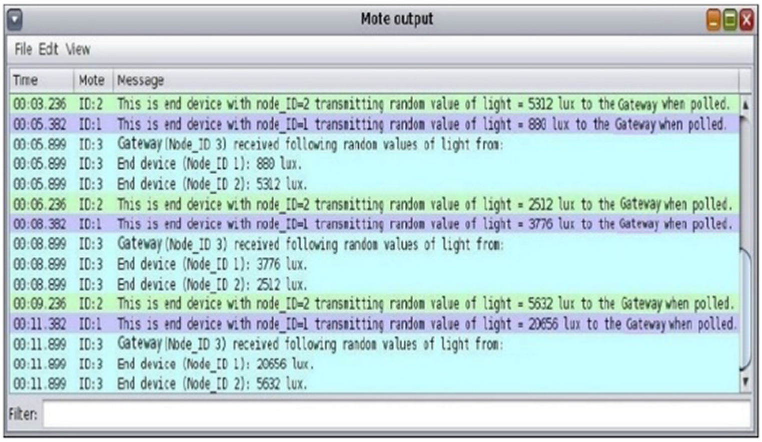
**Fig. 4.** (a) Screenshot of the 'Network window' depicting a 3-node multi-hop network (in operation) within Cooja; (b) Screenshot of the 'Mote output window' obtained from Cooja for the multi-hop network [4].

topological behavior exhibited by the re-orchestrated virtual network are depicted within figures _a and b respectively.

This example case of network re-orchestration performed at the virtual level demonstrates the implications of subjecting a single node within a network, on the overall dataflow or topology of the network in some cases. Such re-orchestrations could prove to be handy towards resolving network fragmentations caused by departure of mobile node beyond the communication range of the gateway. It also aptly implies the efficacy of adopting a software-defined approach (i.e. one involving separation of data plane from



(a)



```
Time       Mote  Message
00:03.236  ID:2  This is end device with node_ID=2 transmitting random value of light = 5312 lux to the Gateway when polled.
00:05.382  ID:1  This is end device with node_ID=1 transmitting random value of light = 880 lux to the Gateway when polled.
00:05.899  ID:3  Gateway (Node_ID 3) received following random values of light from:
00:05.899  ID:3  End device (Node_ID 1): 880 lux.
00:05.899  ID:3  End device (Node_ID 2): 5312 lux.
00:06.236  ID:2  This is end device with node_ID=2 transmitting random value of light = 2512 lux to the Gateway when polled.
00:08.382  ID:1  This is end device with node_ID=1 transmitting random value of light = 3776 lux to the Gateway when polled.
00:08.899  ID:3  Gateway (Node_ID 3) received following random values of light from:
00:08.899  ID:3  End device (Node_ID 1): 3776 lux.
00:08.899  ID:3  End device (Node_ID 2): 2512 lux.
00:09.236  ID:2  This is end device with node_ID=2 transmitting random value of light = 5632 lux to the Gateway when polled.
00:11.382  ID:1  This is end device with node_ID=1 transmitting random value of light = 20656 lux to the Gateway when polled.
00:11.899  ID:3  Gateway (Node_ID 3) received following random values of light from:
00:11.899  ID:3  End device (Node_ID 1): 20656 lux.
00:11.899  ID:3  End device (Node_ID 2): 5632 lux.
```

(b)

**Fig. 5.** (a) Screenshot of the 'Network window' depicting the re-orchestrated 3-node star network (in operation) within Cooja; (b) Screenshot of the re-orchestrated 'Mote output window' obtained from Cooja for the re-orchestrated star network [4].

the control plane) for conduction of soft-trials within a virtualization environment prior to ensuing upon complex network re-orchestrations at the physical level.

## 4.2 Further Complexity in WSN Re-orchestration Scenarios

WSNs is required to be capable of re-assuming their behavior through re-orchestrating the network through both node functional behavior and network-topological standpoints in order to cope with the dynamic 'service' or 're-orchestration' demands in a satisfactory way. Software-defined network re-orchestrations could pave the way for a host of topological adaptations, as illustrated within Fig. 6. These examples reflect the possible formations through the manipulation of the core functions outlined in our work [4].

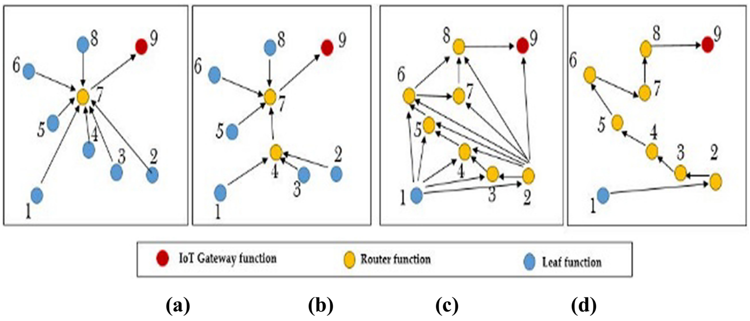


**Fig. 6.** Typical example network topological adaptations for a given WSN as a result of SD re-orchestration (a) Star Topological arrangement; (b) Tree Topological arrangement, (c) Mesh Topological arrangement and (d) Multi-hop Topological arrangement respectively [4].

As a means to determine the improvement in the performance of a network upon undergoing network re-orchestration, consider a scenario wherein a network, initially configured to operate as a multi-hop network (as depicted in Fig. 6d) is re-orchestrated via software control to operate as a star network (as depicted in Fig. 6a) within the virtual (Cooja-based) network [4]. By means of considering 'packet loss' as a performance measure and varying the transmission rate, the performance of the two topological arrangements are compared. Results of the simulation experiment are presented within Table 1.

**Table 1.** Effect of variation of packet communication rate on packets lost for multi-hop and star topology cases (for the same network [4]).

| Packet communication rate | Packets dropped | | | |
|---|---|---|---|---|
| | Star Topology (TDMA) | Star Topology (CSMA) | Multi-hop Topology | |
| 1 PPS | 0 Packets | 0 Packets | 0 Packets | |
| 5 PPS | 0 Packets | 0 Packets | 3 Packets | |
| 10 PPS | 0 Packets | 0 Packets | 5 Packets | |
| 15 PPS | 0 Packets | 0 Packets | 8 Packets | |

From Table 1, it can be seen that as the transmission rate is increased, network tends to lose far fewer packets when operating as 'star' network as opposed to when operating as a multi-hop network. Moreover, implementation of TDMA scheme for the re-orchestrated star network results in no packet losses (at least up to '25' samples per second) whereas four packets are lost for the same when operating under the CSMA scheme.

This simple example too, highlights the significance of conduction of soft trials within the virtual environment to foresee the implications of network re-orchestrations prior to implementation onto the actual hardware nodes present in the physical (OT) layer.

## 5 WSN Re-orchestration Downtime

The process of sensor network re-orchestration can largely be said to encompass the three separate stages of ‘Data Analysis’ and ‘Event Identification’, ‘Planning’ and finally the ‘Implementation or ‘Execution’. The latency of the re-orchestration process as well as the actual downtime suffered by the network when undergoing re-orchestration are identified to be important performance measures that necessitate requisite analyses via experimentation [4].

For this purpose, an example case of network fragmentation caused due to departure of a router node was elaborately detailed within [4] and has been briefly discussed here. The virtual representation of the network is as shown in Fig. 7.

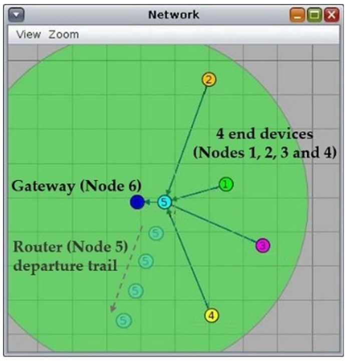


**Fig. 7.** Virtual representation of a 6-node network due to undergo network re-orchestration (owing to fragmentation caused by departure of router node i.e. node 5) [4].

Based on a dedicated knowledge component set apart for constantly monitoring the ‘radio signal strength’ between the gateway node and the router node, a trigger is raised during the initial stage of ‘Data Analysis’ and ‘Event Identification’. This results in initiation of the second stage of the re-orchestration process i.e. ‘Planning’ stage wherein another dedicated knowledge component (as per a suitable fitness model), gathers the requisite information form the underlying layer and determines the most suitable leaf node among all the participant leaf nodes (the ones capable of assuming the functional role of a router and act as a cluster-head for the all remaining leaf nodes). Finally, the ‘execution’ stage involves the implementation of the ‘re-orchestrations’ foreseen within the ‘planning’ stage.

Figure 8 below presents the various messages that are exchanged between the nodes for all the three stages of re-orchestration [4].

Based on the results obtained within Cooja, it was found that the actual ‘downtime’ suffered by the network as a result of the re-orchestration process was equal to that entailed by ‘six messages’, although the overall re-orchestration process entailed much

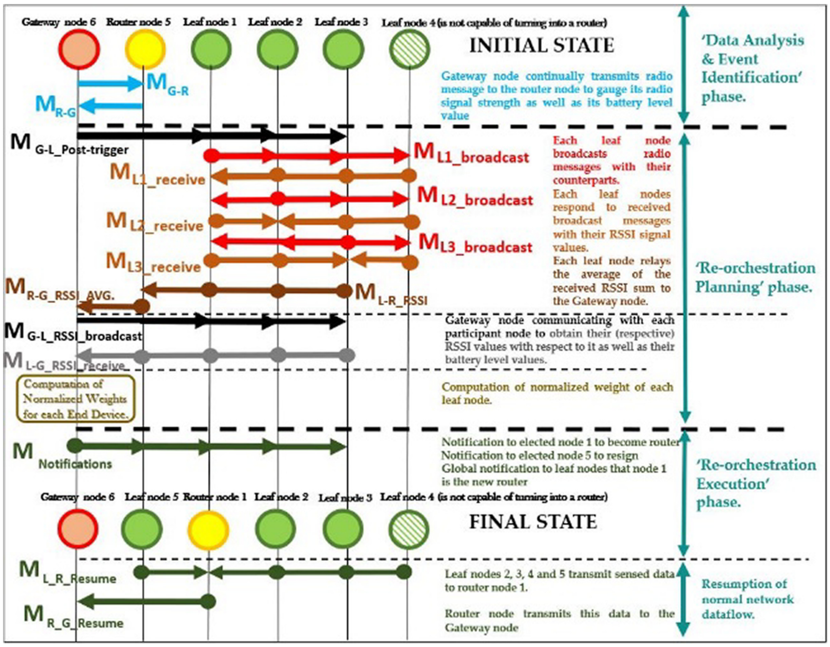

Gateway node 6
Router node 5
Leaf node 1
Leaf node 2
Leaf node 3
Leaf node 4 (is not capable of turning into a router)
INITIAL STATE
Gateway node continually transmits radio message to the router node to gauge its radio signal strength as well as its battery level value
'Data Analysis & Event Identification' phase.
$M_{G-R}$
$M_{R-G}$
$M_{G-L\_Post-trigger}$
$M_{L1\_broadcast}$
$M_{L1\_receive}$
$M_{L2\_broadcast}$
$M_{L2\_receive}$
$M_{L3\_broadcast}$
$M_{L3\_receive}$
$M_{R-G\_RSSI\_AVG.}$
$M_{L-R\_RSSI}$
Each leaf node broadcasts radio messages with their counterparts.
Each leaf nodes respond to received broadcast messages with their RSSI signal values.
Each leaf node relays the average of the received RSSI sum to the Gateway node.
'Re-orchestration Planning' phase.
$M_{G-L\_RSSI\_broadcast}$
$M_{L-G\_RSSI\_receive}$
Gateway node communicating with each participant node to obtain their (respective) RSSI values with respect to it as well as their battery level values.
Computation of Normalized Weights for each End Device.
Computation of normalized weight of each leaf node.
$M_{Notifications}$
Notification to elected node 1 to become router
Notification to elected node 5 to resign
Global notification to leaf nodes that node 1 is the new router
'Re-orchestration Execution' phase.
Gateway node 6
Leaf node 5
Router node 1
Leaf node 2
Leaf node 3
Leaf node 4 (is not capable of turning into a router)
FINAL STATE
$M_{L\_R\_Resume}$
$M_{R\_G\_Resume}$
Leaf nodes 2, 3, 4 and 5 transmit sensed data to router node 1.
Router node transmits this data to the Gateway node
Resumption of normal network dataflow.

**Fig. 8.** Messages exchanged among the various constituent nodes of the network during the three stages of network re-orchestration [4].

higher latency. As mentioned in [4], such results tend to be highly relative in nature. For a more accurate estimate of the downtime caused due to re-orchestration, factors such as the MAC protocol implemented, influence of the dynamics of the physical world on the communication amongst the constituent nodes, etc. need to be considered.

## 6 Example Network Functions Implementation

As a means to offer better elucidation on the three core WSN functionalities of Gateway function, Leaf function and router functions, this section puts forth the pseudo code associated with each of them. These (Contiki-pertinent) Pseudo codes (written specifically in regard to configuration of the Texas Instruments CC2538 microcontroller) outline sample operational activities encompassed by each of the functionalities in a logical way, thereby laying the ground for the functional execution of each, as a whole. While each of the operational activities specified within the three pseudo codes expressed within the subsequent sub-sections can be tweaked as desired via 'Contiki-software' control, certain of them viz., MAC protocol, 'sampling' rate, etc. have been accessed and manipulated via Contiki software for performance evaluation purposes (in regard to the above network re-orchestration scenario i.e. to observe the implications of doing so on network re-orchestration latency) presented within the latter sub-sections.

- ***Selection of requisite sensor(s):*** Select one or more sensor variables (e.g. RSSI, temperature, light, etc. via Contiki)

```
    light=adc_sensor.value(ADC_SENSOR_ALS);
    temperature = adc_sensor.value(ADC_SENSOR_TEMP);
    rssi=packetbuf_attr(PACKETBUF_ATTR_RSSI);
```

- ***Buffering:*** Storing the data sensed within an array say, ‘k’ before transmission

```
     k[0]=Node_ID_number;
     k[1]=light;
     k[2]=temperature;
     k[3]=rssi;
```

***- Communication Addressing type:***
Activating one of the three communication addressing types viz., Broadcast, Multicast or Unicast.

***- Channel Allocation:***

```
#ifndef CC2538_RF_CONF_CHANNEL
#define CC2538_RF_CONF_CHANNEL  <Any channel value from ‘11’ to ‘26’>
```

***- Communication scheme:***
Enabling any of the communication protocols viz., TDMA, CSMA, etc.

*-**Radio Frequency (Output) Transmission Power:***
Set the desired (hexadecimal) value within the requisite ‘*cc2538_rf_power_set*’ function via contiki

```
    cc2538_rf_power_set(uint8_t new_power)
    {
      REG(RFCORE_XREG_TXPOWER) = new_power;
      return (REG(RFCORE_XREG_TXPOWER) & <hex_value>);
    }
```

- ***Setting the Transmission rate:*** Set the desired value (say,) ‘r’ within the ‘etimer’ i.e. ‘event timer function’ via Contiki

```
    etimer_set(&et, CLOCK_SECOND*r);
```

***- Transmission (of sensed data):*** Transmission of the array to requisite node acting as a ‘sink’ node.

```
    packetbuf_copyfrom(&<array_name>, sizeof(<array_name>);
    broadcast_send(&bc);
```

-***Radio Duty Cycling:*** Manipulation of the low power mode (i.e. ‘LPM’ function within Contiki) to configure the Contiki- ported CC2538 to run on one of the following four power modes viz., ‘PM0’, ‘PM1’, ‘PM2’ and ‘PM3’.
    }

**Fig. 9.** Pseudo code for ‘Leaf function’.

## 6.1 Leaf Function Pseudo Code

The pseudo code expressed for the leaf function in Fig. 9 highlights the execution of its core intrinsic operational activities pertaining to ‘sensing’, ‘data acquisition’, buffering, data communication, etc. in a logical way. It commences with the flexible select-ability of one or more sensors (e.g. ‘ambient light’, ‘temperature’, or ‘RSSI’) available at disposal by means of retaining the requisite ‘sensing function’ within the code (and disregarding the ones not needed). The various sensing functions have been specified corresponding to the ‘ambient light’, ‘temperature’, or ‘RSSI variables have been specified within the pseudo code. ‘Buffering’ of the single or multiple sensor variables so sensed involves declaration of an array of certain size, say ‘k’, depending upon the quantity of the sensed variables to be stored. Prior to data communication of the sensed variables stored within the array so declared, parameters such as ‘the communication addressing type’, ‘channel allocation’, ‘communication scheme’ as well as the ‘output radio transmission power’ must be configured. Herein, firstly, one of the three communication addressing types viz., ‘broadcast’, ‘multicast’ or ‘unicast’ is selected and activated. This is followed by selecting one of the channels (15 in total for the TI CC2538 SoC transceivers) available to be accessed (via specifying the number corresponding to the ‘macro’ specified within the pseudo code). The communication scheme could then be realized by means of requisite conditional statements followed by configuration of the ‘output transmission’ power (via specifying the hexadecimal value corresponding to one of the 13 ‘power output values’ available for selection. Finally, the desired rate of transmission could be specified within the ‘etimer_set’ function as a numerical value prior to transmitting it in accordance with the ‘communication addressing type’ (e.g. ‘packetbuf_copyfrom’ function along with ‘broadcast_send(&bc)’ function, when employing the ‘broadcast’ ‘communication addressing’ type).

## 6.2 Router Function Pseudo Code

As outlined within the pseudo code for the ‘router function’ shown in Fig. 10, declaration and initialization of requisite arrays of certain sizes, (say ‘n’,) to accommodate for the incoming data emanating from the group of ‘leaf’ nodes governed by it forms the first part of the router node program. By virtue of the ‘broadcast_recv’ function, the data reported by ‘x’ nodes within its cluster are accumulated within the respective arrays within the router node. Another array ‘d’ (of size equal to the number of values to be transmitted) is declared firstly for aggregation of the data received by the leaf nodes as well as transmission over to the Gateway, router or sink node. As stated within the Subsect. 6.1, certain parameters viz., the type of ‘communication addressing’, ‘channel’, ‘scheme of communication’ as well as the ‘power of output transmission’ can be configured through software control (as explained in Subsect. 6.1). Subsequently, the transmission rate can be easily adjusted by specifying the value in the ‘etimer_set function’ before utilizing the ‘packetbuf_copyfrom function’, together with the ‘broadcast_send(&bc)’ function), to transmit the array consisting of values to be transmitted (i.e. ‘d’ in this case) whilst employing ‘broadcast mode’ of communication (Fig. 10).

```
Initialization of Arrays:
    Declaring and initialization of arrays of size say, 'n' so as to accommodate for incoming sensed
    data from 'n' nodes

     light_v[n]          ={0_1,0_2,….0n};
     temperature_v[n] ={0_1,0_2,….0n};
     rssi_v[n]             ={0_1,0_2,….0n};

    Reception of incoming sensed variables:

      -Reception and storage of sensed data

      int16_t *datapointer_tem;
      datapointer_tem= (int16_t *)packetbuf_datapointer();
      x =datapointer_tem[0];
      light_v[x]           =datapointer_tem[1];
      temperature_v[x] =datapointer_tem[2];
      rssi_v[x]            =datapointer_tem[3];

    - Aggregation of received sensed data within an array 'd'

     d[0]= node_ID_number;
     d[1]= light_v;
     d[2]= temperature_v;
     d[3]= rssi_v;

-    Communication Addressing type:
    Activating one of the three communication addressing types
    - Broadcast
    - Multicast
    - Unicast

-  Setting the MAC Protocol:
                           -CSMA
                           -TDMA

    FORWARDING OF RECEIVED DATA:

    Setting the Transmission rate: Set the desired value (say,) 'r' within the 'etimer' i.e. 'event
timer function' via Contiki

    etimer_set(&et, CLOCK_SECOND*r);

    - Transmission (of sensed data):      Transmission of the array to requisite node acting as a
'sink' node.

    packetbuf_copyfrom(&<array_name>, sizeof(<array_name>);
    broadcast_send(&bc);
    }
```

**Fig. 10.** Pseudo code for 'Router function'.

## 6.3 Gateway Function Pseudo Code

The code for the gateway function too commences with both ‘declaration’ and ‘initialization’ of the arrays to receive and store the incoming sensed data, as present with the pseudo code for the same (shown in Fig. 11). Upon reception, the data values are firstly assigned within another array (declared along with the reception arrays) prior to being subjected to ‘protocol-conversion’ (e.g. 802.15.4 or Zigbee to an IP-based protocol) []. This enables the gateway to ‘escalate’ the sensed data received by it over to a (remote) server over internet. Besides, the gateway also ensues upon ‘management’ (processing, edge-computing, filtering, compression, etc.) of the ‘upstream’ data or exercising some form of control over the ‘downstream’ commands [18, 19].

```
Initialization of Arrays:
    Declaring and initialization of arrays of size say, ‘n’ so as to accommodate for incoming sensed da-
ta from 'n' nodes

        light_v[n]          =$\{0_1,0_2,\ldots.0n\}$;
        temperature_v[n] =$\{0_1,0_2,\ldots.0n\}$;
        rssi_v[n]           =$\{0_1,0_2,\ldots.0n\}$;

Reception of incoming sensed variables:

        -Reception and and storage of sensed data

        int16_t *datapointer_tem;
        datapointer_tem= (int16_t *)packetbuf_datapointer();
        x =datapointer_tem[0];
        light_v[x]          =datapointer_tem[1];
        temperature_v[x] =datapointer_tem[2];
        rssi_v[x]           =datapointer_tem[3];

      - Aggregation of received sensed data within an array 'd'

        d[0]= node_ID_number;
        d[1]= light_v;
        d[2]= temperature_v;
        d[3]= rssi_v;

Conversion of Protocol:
                              - From 'Zigbee' to 'TCP/IP' &
                              - From 'HTTP' to 'CoAP' proxy

Management of 'dataflow' and 'commands':
            -management of flow of 'data'
             -Processing of the 'downstream' commands
```

**Fig. 11.** Pseudo code for ‘Gateway function’.

## 6.4 Factors Influencing Network Re-orchestration Latency

Parameters such as the ‘data communication rate’, ‘total number of nodes’, total numbers of messages exchanged amongst the various (requisite) nodes, MAC communication scheme employed, duration of the time slot (if TDMA scheme has been implemented for the network), contributes to the overall latency. In a bid to gain an estimate the 'extent' to which such parameters tend to influence network re-orchestration latency, experiments could be conducted within the virtual platform of Cooja. One such experiment involving variation of data communication rate has been outlined within the sub-section below.

This experiment has been conducted in furtherance the re-orchestration example case considered in [4] (briefly described within Sect. 5) wherein a network of 6 nodes suffers fragmentation owing to the departure of the router node (node 5) as depicted in Fig. 7.

## 6.5 Effect of 'Data Communication Rate' on 'Re-orchestration Latency'

As a means to determine the effect of variation of the 'Data communication rate' of the constituent nodes on the overall network re-orchestration latency, certain factors viz., network topology, number of nodes, communication scheme (TDMA, CSMA, etc.) were kept constant (This experimental scenario however was trialed for both the communication schemes (i.e. TDMA and CSMA independently).

**Table 2.** Effect of varying sampling rate on Overall Re-orchestration Latency.

| Data communication rate (Packets Per Second i.e. PPS) | Overall Re-orchestration Latency (Star Topology) | |
|---|---|---|
| | CSMA (Seconds) | TDMA (Seconds) |
| 10 | 0.1 | 0.398 |
| 8 | 0.125 | 0.497 |
| 5 | 0.2 | 0.798 |
| 4 | 0.25 | 0.997 |
| 2 | 0.5 | 2 |
| 1 | 1 | 4 |
| 0.75 | 1.33 | 5.329 |
| 0.5 | 2 | 8 |
| 0.25 | 4 | 15.997 |
| 0.2 | 5 | 19.998 |

Table 2 clearly indicates that increase in transmission rate results in decrease in the overall re-orchestration latency and vice versa, for both TDMA and CSMA cases. For lower values of data communication rates, overall re-orchestration latency tends to be significantly higher when TDMA scheme is implemented for the network (as opposed when CSMA is adopted for the same).

## 7 Conclusion

A cyber-physical architectural framework (consisting of reusable modular functionalities) that allows for virtualization and testing of the underlying physical sensor network is deemed viable towards realizing the objective of SDWSN. The approach pertaining to 'conditional execution' of the functional modules (pre-defined within the code with which the constituent resource-rich physical transceivers are configured) by means of requisite external radio signal messages is one of the ways to achieve flexible re-orchestration and has been adopted for this work. Several other approaches, most notably, OTAP-based approaches or fuzzy logic-based approaches, could be adopted to attain the objective of SDWSN in a more effective manner. Re-orchestrations occurring within such SDWSNs tend to be accompanied by some amount of 'network downtime' that may prove to be partially or massively detrimental to its ongoing operational processes. Such 'downtime' tends to be dependent on certain network parameters like 'data communication rate', 'number of nodes, etc. within the tree structure of the SDWSN.

## References

1. Acharyya, I., Al-Anbuky, A.: Towards wireless sensor network softwarization. In: IEEE NetSoft Conference and Workshops NetSoft, Seoul, Korea Republic, pp. 378–383 (2016)
2. Ezdiani, S., Acharyya, I., Sivakumar, S., Al-Anbuky, A.: Wireless sensor network softwarization: towards wsn adaptive QoS. In IEEE Internet of Things Journal **4**(5), 1517–1527 (2017)
3. Acharyya, I., Al-Anbuky, A., Sivakumar, S.: Software-Defined Sensor Networks: Towards Flexible Architecture Supported by Virtualization. In: Global IoT Summit GIoTS, Aarhus, Denmark, pp. 1–4 (2019)
4. Acharyya, I., Al-Anbuky, A.: Software-defined wireless sensor network: WSN virtualization and network re-orchestration. In: Proceedings of the $9^{th}$ International Conference on Smart Cities and Green ICT Systems SMARTGREENS 2020, LNCS, vol. 1, pp. 79–90 (2020)
5. Krasteva, Y., Portilla, J., De la Torre, E., Riesgo, T.: Embedded Runtime Reconfigurable Nodes for Wireless Sensor Networks Applications. In IEEE Sensors Journal **11**(9), 1800–1810 (2011)
6. Eronu, E., Misra, S., Aibinu, M.: Reconfiguration approaches in wireless sensor network: issues and challenges. In: IEEE International Conference on Emerging and Sustainable Technologies for Power and ICT in a Developing Society NIGERCON 2013, Owerri, pp. 143–142 (2013)
7. Aslam, M., Hu, X., Wang, F.: SACFIR: SDN-Based Application-Aware Centralized Adaptive Flow Iterative Reconfiguring Routing Protocol for WSNs. Sensors **17**, 2893 (2017)
8. Huang, M., Yu, B.: Towards general software-defined wireless sensor networks. In: Proceedings of the 4th IEEE International Conference on Computer and Communications ICCC 2018, Chengdu, China, pp. 923–927 (2018)
9. Oliveira, B., Margi, C.: Distributed control plane architecture for software-defined Wireless Sensor Networks. In: IEEE International Symposium on Consumer Electronics ISCE 2016, Sao Paulo, pp. 85–86 (2016)
10. Kobo, H., Abu-Mahfouz, A., Hancke, G.: A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements. In IEEE Access **5**, 1872–1899 (2017)
11. Kobo, H., Hancke, G., Abu-Mahfouz, A.: Towards a distributed control system for software defined Wireless Sensor Networks. In: In 43rd Annual Conference of the IEEE Industrial Electronics Society IECON 2017, Beijing, pp. 6125–6130(2017)

12. Kgogo, T., Isong, B., Abu-Mahfouz, A.: Software defined wireless sensor networks security challenges. In: IEEE AFRICON 2017, Cape Town, pp. 1508–1513 (2017)
13. Jian, D., Chunxiu, X., Muqing, W., Wenxing, L.: Design and implementation of a novel software-defined wireless sensor network. In: In 3rd IEEE International Conference on Computer and Communications ICCC 2017, Chengdu, pp. 729–733 (2017)
14. Ezdiani, S., Acharyya, I., Sivakumar S., Al-Anbuky, A.: An IoT environment for wsn adaptive QoS. In: IEEE International Conference on Data Science and Data Intensive Systems 2015, Sydney, NSW, Australia, pp. 586–593 (2015)
15. Ezdiani, S., Acharyya, I., Sivakumar S., Al-Anbuky, A.: An Architectural Concept for Sensor Cloud QoSaaS Testbed. In: Proceedings of the $6^{th}$ ACM Workshop on Real World Wireless Sensor Networks RealWSN 2015, pp. 15–18 (2015)
16. Zhou, Y., Lyu, M.R., Liu. J.: On sensor network reconfiguration for downtime-free system migrations. In: Proceedings of the 5th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness QShine 2008. Brussels, Belgium, Article 24, pp. 1–7 (2008)
17. Szczodrak, M., Gnawali, O., Carloni, L.: Dynamic configuration of wireless sensor networks to support heterogeneous applications. In: IEEE International Conference on Distributed Computing in Sensor Systems Cambridge, MA, pp. 52–61 (2013)
18. Baghyalakshmi, D., Kothari, S., Ebenezer, J., SatyaMurty, S.: Ethernet gateway for wireless sensor networks. In: Twelfth International Conference on Wireless and Optical Communications Networks (WOCN), Bangalore, pp. 1–5 (2015)
19. Yuan, Z., Cheng, J.: The Design and Realization of Wireless Sensor Network Gateway Node. Adv. Mater. Res. **760**, 462–466 (2013)