Design and Implementation of a Smart Home Networking Simulation

Cheng Jin and Thomas Kunz

Technical Report SCE-10-05

Department of Systems and Computer Engineering
Carleton University
Ottawa, Canada

August 2010

Abstract

Given the power grid technology, integrated with renewable power generation technologies, Demand-Response (DR) programs enabled by Advanced Metering Infrastructure (AMI) were introduced into the power grid in the interest of both utilities and residents to achieve load balance and improved grid reliability by encouraging residents to reduce their power usage during peak load periods with extra premiums in return. From the perspective of energy saving and power efficiency in smart homes, a cost-effective Home Energy Management System (HEMS) is capable of automatically supervising energy-aware smart appliances, small-scale renewable energy generation facilities and plug-in vehicles around the houses in flexible cooperation with AMI to deliver time-based price messages from utilities to residences. However, lots of emphasis was placed on the energy management of the whole grid and the corresponding underlying communication infrastructures on a large scale mainly due to the unavailability of realistic test-beds and lack of support in existing software simulation environments feasible to smart homes.

Inspired by the approaches of Multiple Interfaces and Multiple Channels (MIMC) in the literature, we designed and implemented a simplified networking simulation model in the Network Simulator Version-2 (NS-2) to explore the execution of DR programs and evaluate the network performance in smart homes. The model includes a Radio Broadcast Data System (RBDS) network and a combination of ZigBee/IEEE 802.15.4 plus HomePlug C&C with the support of multiple communication channels for each network individually. In addition to the node construction as well as the addressing scheme intended for nodes with multiple interfaces, this report mainly focuses on the node organization and management of communication channels, the mechanism of packet tagging through interfaces at the MAC/PHY layer, the implementation logic of routing protocols featuring various routing strategies associated with specific scenarios and the basic functionalities of the application layer from the bottom up. Also, a comprehensive guideline is presented here to help configure the simulation model with multiple networks specific to scenarios in a smart home.

TABLES OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

Intelligent management of the power grid, aiming at promoting more even utilization of electricity and minimize energy loss during power transmission and consumption is currently highlighted at the global level by utilities, academic organizations as well as public administrations. To protect the interest of both utilities and customers to the full extent, the idea of smart grid coming with enabling technologies has been put forward over recent years and attracts great attention from the power industry and academy engaged in such explorations.

As an emerging technology in terms of electricity grid management, the smart grid [1] integrates electronics and information technologies into the massive electric systems in such a way as to strengthen reliability, flexibility, security, safety and efficiency as a whole. Technically speaking, it employs innovative devices combined with machine automation, remote-control and wireless sensors through underlying communication infrastructures. From the perspective of economy and energy saving, it minimizes the electricity consumption during expensive peak hours by coordinating the load balance in the systems and leveraging demand-response mechanisms with time-based pricing notification oriented towards residents so as to provide consumers with high-quality and cost-effective services.

With the deployment of smart grids, it is feasible to integrate renewable power generation technologies originated from different sources into the power grid for the purpose of coordination of energy utilization in the event of a power crunch. In this way, the smart grid technology manages to meet the ever-increasing demand of minimizing the negative impact upon the environment while achieving high performance.

One of the aspects with regard to power grid management that electricity utilities are confronted with is effective and smart approaches to cope with peak load as well as other emergencies regardless of their occurrences. Considering the infrequency and short periods under such circumstances, a Demand-Response (DR) program [2], as one of the most common services in smart grid technology, has been introduced into the power grid. In this way, utilities are capable of achieving load balance in the power grid through the DR procedure by encouraging customers to reduce their electricity consumption during peak load periods with special bonus/incentives in return. Meanwhile, residents could benefit from the DR services in terms of the electricity bill reduction when adjusting their electricity usage of home appliances in houses in response to dynamic pricing and other events associated with the reliability of the power grid issued by utilities.

Acting as a key enabling technology, the Advanced Metering Infrastructure (AMI) [3][4] has been widely deployed to facilitate DR programs in smart grids. Generally, AMI covers smart meter units, device networking infrastructures, communication technologies, network management platforms as well as integration frameworks. With the support of AMI, a time-varying price notification is delivered by utilities to smart meters located in residents' houses. As a consequence, smart meters forward the signal to home devices intelligently configured in houses by communicating with them in a wireless or wired way so as to accomplish end-to-end pricing transfer and power usage adjustment intended for home devices.

In addition to the AMI-enabled DR program advocated by utilities, effective energy management within smart homes also has to be taken into account in the context of underlying infrastructures in smart grids. From the perspective of energy saving and improvement in power efficiency, a platform-centralized Home Energy Management System (HEMS) [5] plays a key role in automatic supervision of energy-aware smart appliances, small-scale renewable energy generation infrastructures around the houses and plug-in vehicles and flexible cooperation with AMI in delivering resident-oriented messages from utilities.

Therefore, it is interesting to explore what kind of networking technology suits well with smart homes and the efficiency of message transfer along with network performance in the smart home network in conjunction with the advancement of smart grid technology. On the one hand, there are few publications and articles discussing such topics, particularly in terms of energy saving control due to the unavailability of realistic test-bed with a reasonable scale for independent experiments and lack of support in software simulation environments feasible to facilitate such evaluations; on the other hand, lots of emphasis was placed on the energy management from the scope of the whole grid and corresponding underlying communication infrastructures on a large scale in the interest of utilities.

To fill in the gap and boost research on energy management and networking technology involved in smart homes, we designed and implemented an experimental model in the Network Simulator Version-2 (NS-2) software simulation environment with resources publicly available to us in an effort to evaluate the networking issues involved in smart homes.

The remaining part of this report is organized as follows: Chapter 2 gives a brief introduction to the features of the smart home framework and the simulation model we proposed and accomplished in NS-2. Chapter 3 reviews the characteristics of the NS-2 software framework that deeply influences the node construction and data communication in the case of multiple interfaces/channels. Chapter 4 summarizes the existing approaches to multiple interfaces/channels specific to networking scenarios and describes our solution that fits in our simulation model, given the operational mechanism and node construction in NS-2. Chapter 5 describes the node management in the software communication channel, modifications involved in our model as well as the issue of interference in terms of packet scheduling. Chapter 6 presents different MAC/PHY protocol stacks, ranging from RBDS and HomePlug C&C to ZigBee/IEEE 802.15.4 in terms of data packet tagging and features in data transmission. Chapter 7 focuses on the routing strategies of data packet forwarding in multiple networks, mainly emphasizing the implementation logic adopted in Ad hoc On Demand Distance Vector (AODV) routing, ZigBee routing and Flooding, which are considered to be competitive candidates to the multi-network scenario in smart homes. Chapter 8 features the basic functionalities at the application/transport layer in NS-2. Chapter 9 highlights the addressing issues that occur in data transmission among nodes with multiple interfaces/channels. Chapter 10 provides a comprehensive guideline of how to configure nodes with multiple interfaces/channels in establishing the simulation model along with specific situations during execution.

2 Background and Simulation Model

Initially originated in the US, home automation [6] is designed to employ microcontrollers to monitor and adjust such home appliances as electrical ovens, water heaters, dishwashers, washing machines and dryers, indoor lighting, refrigerators as well as Heating/Ventilation/Air-Conditioning (HVAC) facilities in terms of temperature or humidity in response to the home owner's requirements. To some extent, home automation is partially in charge of the indoor power consumption with the instructions of household owners. However, home automation mostly represents the proactive and local control on the basis of the preference of house owners rather than on the basis of the whole power grid at a higher level in terms of energy utilization and load balance. For this reason, smart home technology is shown as a reasonable alternative to energy management in place of home automation in houses in the interest of both electricity utilities and household owners.

As an integrated system, a smart home [7] makes the utmost of a range of techniques from a centralized platform for management and control, intelligence devices, underlying communication infrastructures to synthesized wiring in such a way as to connect all indoor subsystems with home appliances and household electrical devices attached as a whole. Specifically, smart home enabling techniques enable householders to shift from simple independent control to effective centralization of management and services in a house, providing them with all-round features for internal information exchange and helping to keep in instant contact with the outside world. In terms of convenience, they help people in optimizing their living style, rearranging the day-to-day schedule, securing a high quality of living condition and in turn enable people to reduce power bills from a variety of energy consumptions in a house.

In addition to the management of energy consumption by home devices, a smart home with the support of HEMS is also capable of controlling the operations of distributed power generation facilities including solar panels and wind turbines, as well as the charging of PEV/PHEV as the standby power source available, so as to relieve any power constraints during peak load periods and unexpected outages or blackouts. In this way, the smart home technology bridges between a stand-alone home network and supporting infrastructures in smart grids in terms of power load supervision and adjustment.

In the context of smart grid management, AMI mainly takes responsibility of delivering messages issued by utilities to each house, distributed in all areas over long-distance power line or wireless communication mediums, as Figure 1 illustrates:

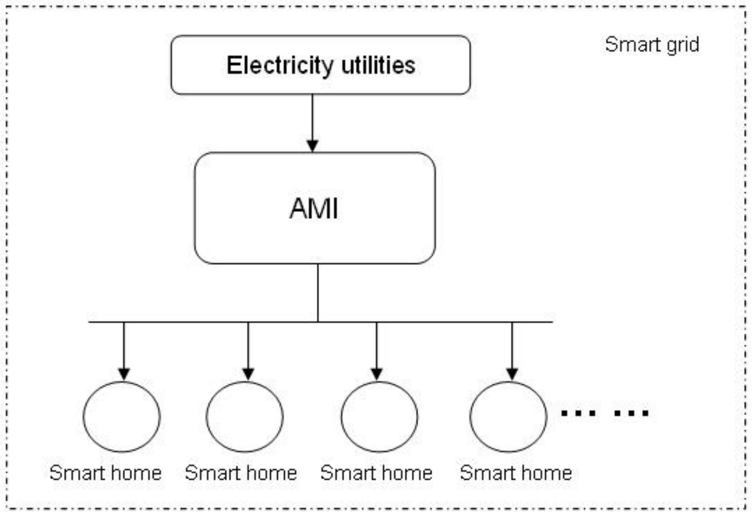


Figure 1 Scenario of time-varying price through AMI to smart homes in smart grids

Inside a smart home, a smart meter deployed by utilities keeps track of message originated from utilities through AMI and cooperates with the household central controller so as to schedule the usage of energy for home appliances based on the preference and pre-configuration of residents.

Upon reception of messages from the smart meter, the central controller selectively adjusts power supply and energy consumption oriented towards home appliances in a house. First of all, it queries the accessibility of energy generation and storage facilities available for the moment. Those facilities involved could automatically switch their output towards the residence upon request without further intervention from the controller, with the power grid as a supplement. In the event of a shortage of power supply among those facilities, the controller immediately sends a postponement message based on price dynamics to all home devices including Plug-in Electric Vehicle /Plug-in Hybrid Electric

Vehicle (PEV/PHEV) featured with high power consumption, regardless of their current operation status or simply cut off power for a couple of devices when urgent.

To simplify our model, we only focus on the connection of AMI with smart homes for message delivery and data transmission within smart homes. Therefore, there are two parts to be taken into consideration: the long-distance communication technology for data transmission from utilities to houses and the networking technologies feasible to be employed in smart homes.

Among technologies available in one-way communication over a considerable distance, Frequency Modulation (FM) provides good support in signal penetration through buildings when broadcasting data to destinations. Radio Broadcast Data System (RBDS, a North American radio broadcast standard equivalent to the Radio Data System in Europe) [8] could be employed for transmission of small-size packets over the FM channel in that it covers most of the residences in North America. In addition, a RBDS-Utility Message Channel (RBDS-UMC) [9] network deployed by e-Radio has already been put into practice in providing message delivery services to utilities and the DR service providers. With the e-Radio Operations Centre as a data forwarding platform, the RDBS network enables utilities to issue electricity-related messages to thermostats, appliance controllers and in-home display units in house units via a radio broadcasting service provided by FM radio stations.

Meanwhile, the authors in [10] have shown in simulations that a RBDS network presents us a feasible alternative to effectively deliver messages to enabling devices intended for DR programs in houses.

With the RBDS-enabled communication model implemented in [10], we choose to make the utmost of the RBDS network model in our research project to simulate the behavior of message transmission through an AMI-enabled demand-response program to smart homes.

With respect to networking technologies in smart homes, both wireless and wired communication protocol stacks have to evaluated and compared based on our requirements in smart homes. On one side, short-distance wireless technologies emerging in recent years are featured with low speed, lower power consumption supported by battery supply, high cost-effectiveness and more flexibility in terms of networking and deployment in a house. However, they mostly suffer issues including mutual interference with other technology transmitting in a shared band, signal attenuation, shadowing and fading as well as multipath effects in the wireless environment that could deteriorate the quality of data transmission. On the other side, Power Line Communication (PLC) [11] is considered superior to other alternatives due to the accessibility of power outlets in each room, which avoids the extra costs of wiring in most residences and thus promotes the convenience of promisingly seamless communication with utilities via power line. Nevertheless, patent restrictions, lower data rate than expected, as well as high costs unaffordable to most residents inherent in some of technologies restrict them in the advancement of smart home networking.

Generally, there is no perfect solution to address every aspect in smart homes based on either PLC technologies or short-range wireless network technologies. From the perspective of energy saving, it is more desirable to combine both wireless network technologies and wireless and PLC technologies in a mutual way so as to satisfy the practical demands in smart homes. Therefore, we propose a networking solution of combining ZigBee/IEEE 802.15.4 with HomePlug C&C that seems promising to smart homes in the sense that other factors are also taken into account in our proposal, such as openness of protocol stacks, layering-based interoperability and cost-effectiveness sensitive to customers, etc. In this combination, a ZigBee/IEEE 802.15.4 network could still work with HomePlug C&C with the support of the renewable energy generation facilities available in the event of a power outage, whereas a HomePlug C&C network functions as an extension and redundancy necessary for data transmission in a house.

In order to evaluate the network traffic metric and energy consumption on sensor-enabled nodes, we design and implement a simplified smart home networking simulation model in NS-2 v2.33 so as to explore the execution of DR programs in smart homes. The complete simulation model includes a RBDS network as well as an indoor networking scheme of ZigBee/IEEE 802.15.4 plus HomePlug C&C in an environment of multiple communication channels intended for each network respectively.

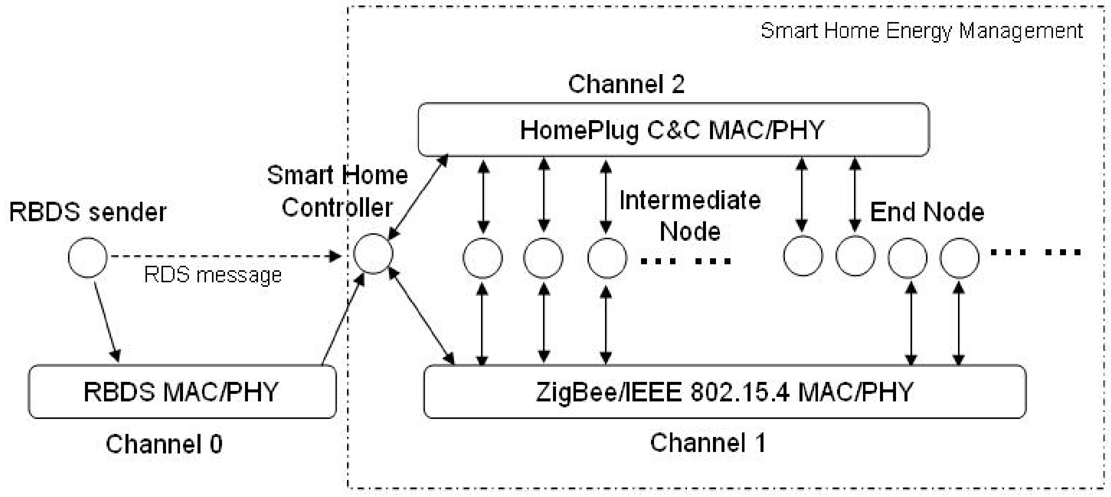


Figure 2 Simulation model of energy control network in smart homes

As illustrated above in the simulation model, two nodes are involved in RBDS message transmission: the RBDS message sender at the electricity utilities (third-party service providers) holds one interface to the RBDS network, and a smart home controller (an integration of a smart meter and a centralized control platform in houses) holds three interfaces to the RBDS network, the ZigBee/IEEE 802.15.4 network and the HomePlug C&C network irrespectively. A group of intermediate nodes are featured with interfaces to both the ZigBee/IEEE 802.15.4 network and the HomePlug C&C network in the house and function as both message recipients and routers mainly targeted for RBDS message forwarding in the network based on the routes initially established to destinations. Also, this type of nodes could partially be configured as required to work through one interface to the HomePlug C&C network in order to mimic the behavior of smart home appliances that are only connected to the power line, given the wireless interface disabled in data transmission. The remaining nodes with one interface to the ZigBee/IEEE 802.15.4 network only act as smart home appliances located anywhere in a house sporadically receiving RBDS messages through the central controller. Also, they could forward RBDS messages among nodes only in the ZigBee/IEEE 802.15.4 network for the purpose of simplicity.

Upon reception of a RBDS broadcast messages from the RBDS sender, the controllers decides to send different messages reframed with new destination addresses and the packet type indentified in a smart home to a range of devices in response to the new information gained about the power status/costs. All nodes are configured in advance to receive the message according to residents' preference. Meanwhile, any sensor-enabled node in the networking combination is capable of directly communicating with the smart home controller any time in terms of feedback of command execution, status report, etc.

Unfortunately, the NS-2 simulation environment is not supposed to be employed in support of multiple protocol interfaces along with different channels originally due to the lack of consideration in terms of realistic applications and software architecture, which inevitably poses a great challenge to all other software simulations similar to our topic. To meet our requirements of smart home networking simulation, a couple of key issues have to be addressed as a whole in the corresponding design and implementation: the composition and dynamic configuration of mobile nodes, node organization and management of communication channels, packet tagging through different underlying interfaces, options of routing strategy that suit well the case of multiple interfaces/channels as well as the mechanism of centralized data forwarding, and so forth.

3 Operational Mechanism of NS-2

NS-2 [12][13] is an open source, Object Oriented (OO), and discrete-event simulation tool mainly used to explore the performance of wired and wireless networks in terms of underlying protocol stacks, routing algorithms, network traffic, etc. The main functionality of NS-2 is to establish a network of nodes that are able to communicate with each other by transmitting or receiving data packets over the network, and subsequently to generate the trace result of network traffic for analysis. To achieve this, NS-2 contains various network-related components that can be flexibly assembled by a descriptive language called Tool Command Language (Tcl) specific to certain networking configurations to allow different simulation scenarios.

NS-2 is chiefly featured with the construction of network components, the split object model on the basis of two kinds of OO language as well as the network-related event scheduling mechanism. These features are summarized as follows.

3.1 Construction of Network Components

All networking components in NS-2 are implemented by taking advantage of the mechanism of inheritance and polymorphism in the OO technology. NsObject is the base network component already defined in NS-2 that is responsible for abstracting generic behaviors of network components such as packet handling and packet delivery. In this way, any other network component (except node) inherited from NsObject extends the basic behaviors in NsOjbect to specific operations under different circumstances.

3.2 Split Model and Variable Binding

Normally, the network simulation includes two steps: defining networking scenarios and executing the simulation. On one hand, all kinds of parameters associated with network components must be configured in advance and are allowed to vary with time during simulations; on the other hand, a system programming language with high performance is required to efficiently cope with processes including packet handling and the execution of various algorithms. Therefore, it is reasonable for the two behaviors to operate independently from one another. Meanwhile, a system pipeline between the configuration environment and the execution environment has to be established in such a way as to transfer the configuration information to objects in the execution environment at the beginning of simulation.

To meet these demands, NS-2 takes advantage of two languages by combining both of them for simulations. One of them is the Object-oriented Tool Command Language (OTcl) that is used to interpret each line of configuration scripts and assemble network components involved after creation; the other is C++ that is employed in implementing or extending specific protocol stacks and corresponding algorithms for packet handling held by network components in simulations. Eventually, both languages generate their own object spaces suitable for later execution, as illustrated in Figure 3.

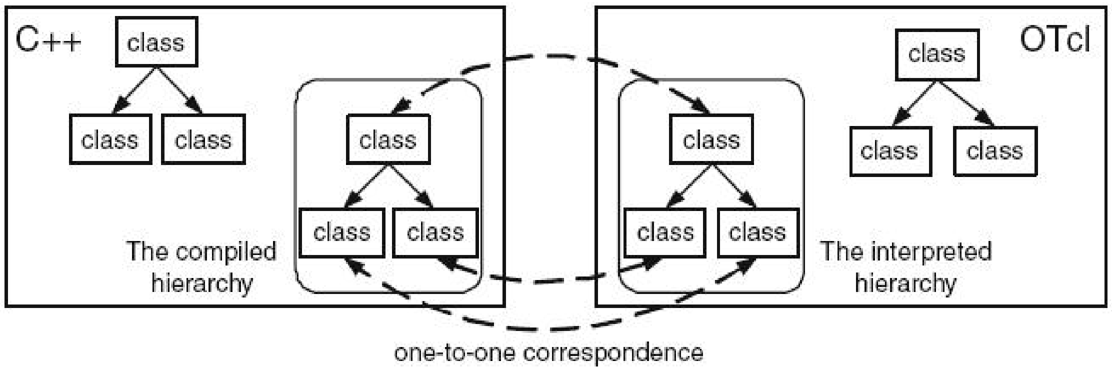


Figure 3 The split model in the scope of NS-2 [13]

To be specific, network objects generated by corresponding classes are organized in a hierarchical way respectively during the initialization of the simulation environment. First of all, the objects in the interpreted hierarchy are generated by the simulator instance. Following that, the corresponding objects in the complied hierarchy (also called shadow object related to the objects in interpreted hierarchy) are created through a static mapping mechanism from OTcl to C++.

Considering that the cross-reference to class variables belonging to a different scope of language is prohibited in NS-2, the class variables existing in both interpreted objects and complied objects have to be bound together so as to synchronize the change of variable in both hierarchies. In terms of the generation process of a network object, the binding action from an interpreted variable to a compiled variable across the hierarchies takes place during the generation of the compiled shadow objects. In such way, the hierarchies of both languages are linked together so as to establish a one-to-one relationship between both interpreted and corresponding complied objects.

A typical example is the generation process of communication nodes, as shown below:

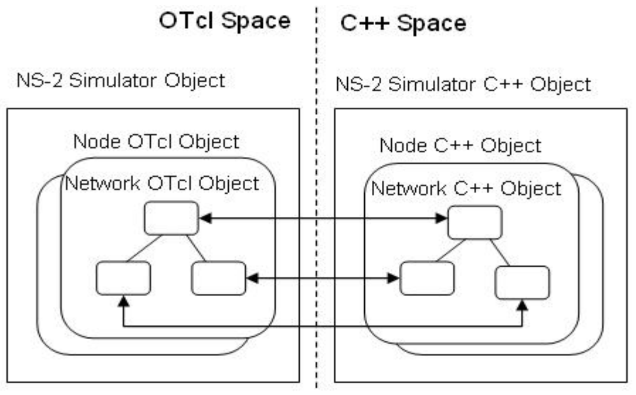


Figure 4 Mapping of network objects based on the split model

After the initialization of the simulation environment, the interpreted object of a simulator in the OTcl domain takes responsibility for the creation of connecting nodes as well as the corresponding networking components such as applications, routing agents, underlying protocol stacks, communication channels, etc. Besides, the shadow objects in the C++ domain are also created through the static mapping. Through the variable binding, any change made in scenario scripts will be transferred from interpreted variables to complied variables and further influences the simulation execution in the C++ domain. Therefore, the order of generation for each node and the setting of corresponding variables have to be considered carefully before establishing a simulation scenario.

3.3 Event scheduling

NS-2 simulations work in an event-driven manner, which means that the data transmission among networking nodes and other time-triggered actions related to network objects are mostly implemented on the basis of the mechanism of discrete event scheduling. Specifically, a scheduler object, along with an event queue, is initialized for a single simulation scenario, as illustrated in Figure 5:

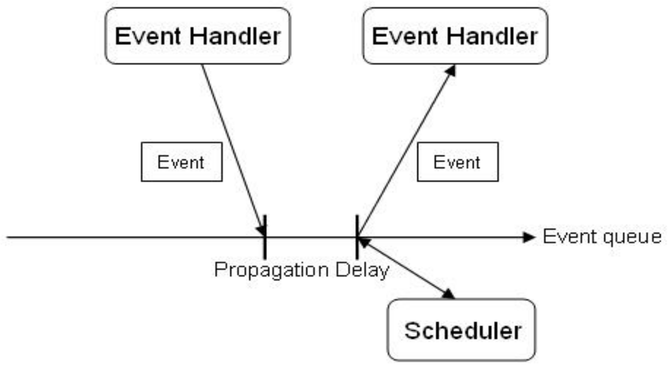


Figure 5 Event scheduling in NS-2

Essentially, an event is treated as a message object issued by a network object or simply a data packet delivered by a protocol layer related object. In addition to the critical data to be handled by the target objects, an event also consists of a unique ID statically assigned by the scheduler object, a timestamp based on the system virtual clock in simulations and the propagation delay calculated from the distance between nodes, as well as an event handler pointing to the target objects.

The scheduler object is responsible for maintaining the event queue by tracking the scheduled time of each event in the queue. By comparing the timestamp in events with the current clock, it chronologically extracts events from the queue and transfers the handling to target objects through a software callback mechanism. Meanwhile, the scheduler object follows the principle of First-In-First-Out (FIFO) intended for queue management in the case that events collide with each other in terms of the scheduled time. Under such circumstances, these events to be dispatched simultaneously are always scheduled in order of precedence. In addition, the uniqueness of an object address in the program space of simulation guarantees that only the right object is invoked to handle the scheduled event.

4 Mobile Node Configuration in NS-2

4.1 Architecture of a Mobile Node

To simulate various scenarios in wireless networks, a group of mobile nodes are configured based on the interpreted hierarchy in NS-2 so as to be created as runtime objects in the compiled hierarchy for data communication. From the perspective of node construction, network objects within a mobile node are linked together through target object pointers in a layered manner, as illustrated below:

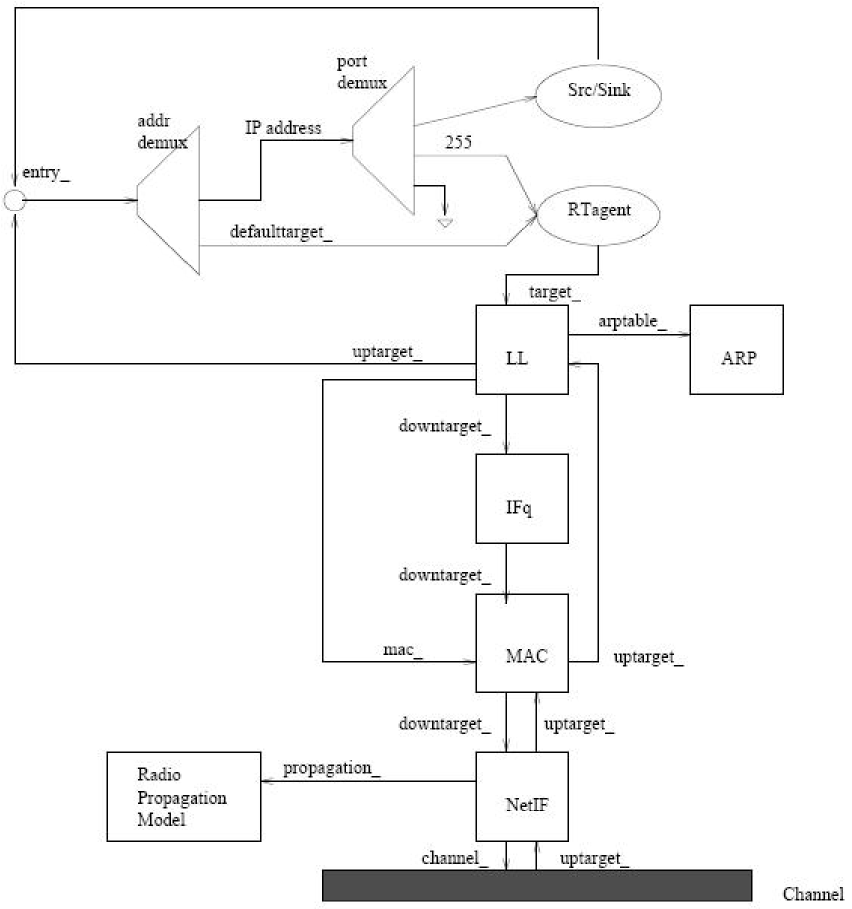


Figure 6 The architecture of a mobile node in NS-2 [12]

The upper layers includes the application part (application source object and transport layer object at the sender side are intended for the generation of data packets while a null agent object at the receiver side simply releases any incoming packet) and a routing layer object(or a routing agent). An address classifier along with a port classifier (or multiplexer) are employed to identify the destination address and the corresponding port number in a packet header when dealing with data packets to be transmitted or received. It is up to the routing agent to forward outgoing packets to destination nodes via the supporting layers.

The supported protocol stack is composed of a Link Layer (LL) object plus an Address Resolution Protocol (ARP) object, an InterFace priority Queue (IFQ) object intended for packet buffering, a Media Access Control (MAC) layer object and a Network InterFace (NetIF or PHY) layer object bound to a wireless propagation model object. For each mobile node in the network, the whole layering framework is connected to the same communication channel object via the PHY layer. After the calculation of communication delay and the range of the transmission signal, mobile nodes are able to communicate with each other by transmitting packets over the same channel with the support of the packet event scheduling mechanism.

It is evident that the architecture of a mobile node and the operational mechanism of network objects within a mobile node already implemented in NS-2 are only suitable for the creation of nodes with a single channel in a single network. To meet our requirements of multiple channels over multiple networks in smart home networking simulations, the connection between network objects, the organization of connecting nodes as well as part of the implementation logic within network objects needs to be revised on the basis of the existing features provided in NS-2, especially the features supported by OTcl in terms of the construction of mobile nodes.

4.2 A Brief Survey of Solutions to Multiple Interfaces/Channels

In order to improve the overall performance and throughput in wireless networks, the topic of Multiple Interfaces and Multiple Channels (MIMC) has already been attracting attention in the academic community. From the perspective of software modeling and simulation, the major efforts are placed upon the creation of mobile nodes equipped with MIMC and the optimization of MAC and routing options aiming at promoting the utilization of multiple channels (discussed in Chapter 7) within a mobile node when transmitting data packets. To serve as a group of constructional comparisons with our implementation, these proposed solutions to MIMC adopted in NS-2 are briefly summarized below on the background of distinct networking scenarios.

4.2.1 TENS

As a framework extension of the existing NS-2 implementation, The Enhanced Network Simulator (TENS) project [14] was built in an attempt to address the omissions encountered in modeling the IEEE 802.11 Wireless Local Area Network (WLAN). To make it more realistic in simulations of long distance links, the TENS project is also characterized with support for multiple interfaces for mobile nodes, a static routing protocol as well as a simplified directional antenna model, as partially illustrated in Figure 7:

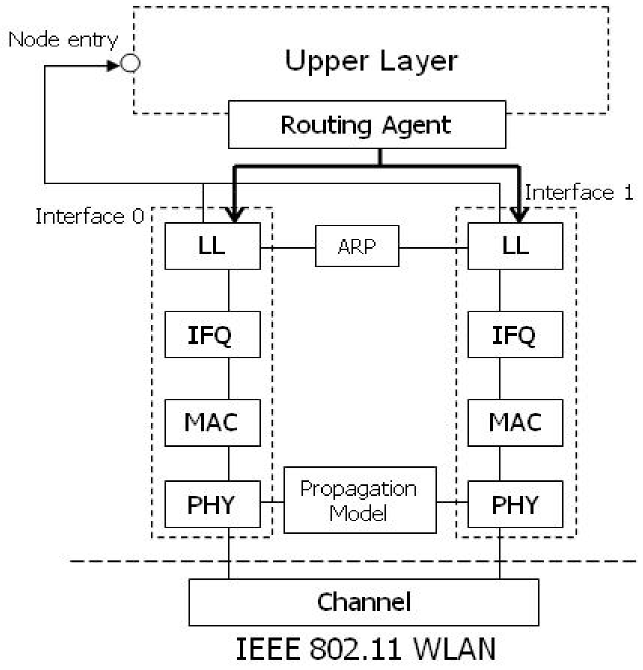


Figure 7 Architecture of mobile node in TENS

Specifically, it is assumed in this solution that all nodes co-exist in only one network regardless of the number of their interfaces. As a consequence, a table for MAC address resolution as well as a radio propagation model is implemented in a shared way among interfaces within a mobile node. Apart from that, a single channel object is multiplexed in the architecture to mimic the packet delivery behavior over multiple channels by indexing each pseudo-channel explicitly in scenario scripts. With the shared channel object, the metric associated with co-channel interference could be statically calculated for each PHY layer object when transmitting packets in the same network.

In terms of the construction of a mobile node, this solution does not differentiate which interface at a receiver side is eligible to handle incoming packets. Hence, the PHY layer object bound with one pseudo-channel within a mobile node could receive packets originated from other pseudo-channels with high probability. In addition, the mechanism of simplified channel multiplexing downgrades the realistic behaviors occurring in multiple channels to some extent and further impacts the network traffic with the increase of mobile nodes in that all packets to be transmitted in the network are scheduled simultaneously over the same channel object.

4.2.2 Hyacinth

A multi-channel Wireless Mesh Network (WMN) architecture called Hyacinth was proposed by the authors in [15] to effectively address the issue of constrained bandwidth by fully utilizing existing non-overlapped radio channels available in the IEEE 802.11 specifications. The Hyacinth prototype was evaluated in NS-2 simulation and a test bed experiment environment was also established to demonstrate its feasibility in improving the network throughput as compared to a single channel ad hoc network when the prototype was deployed in an wireless network owned by an Internet Service Provider (ISP) or a wireless enterprise backbone network, as illustrated in Figure 8.

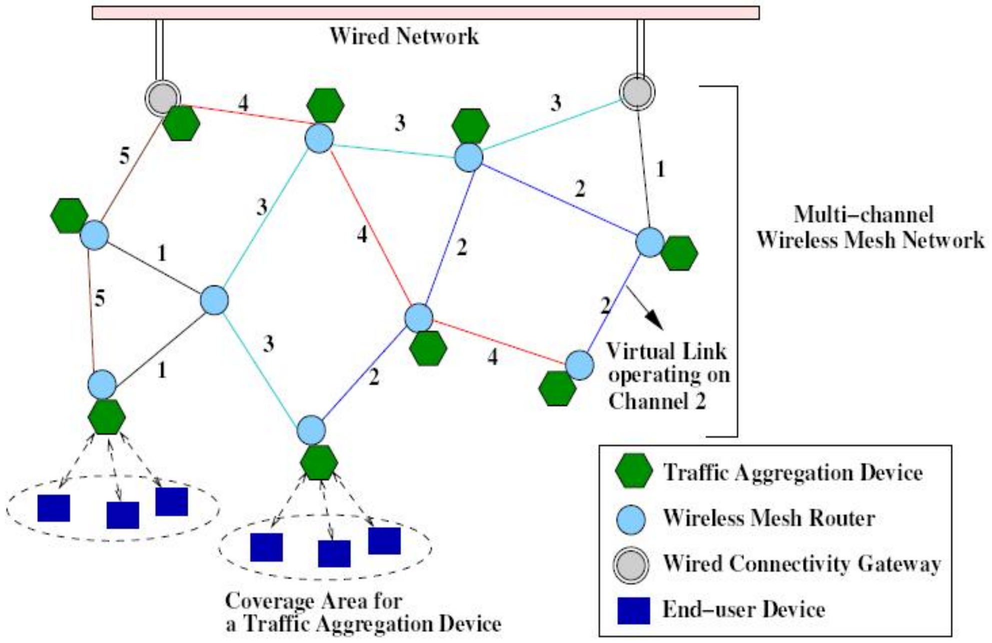

Wired Network
Multi–channel Wireless Mesh Network
Virtual Link operating on Channel 2
Coverage Area for a Traffic Aggregation Device
Traffic Aggregation Device
Wireless Mesh Router
Wired Connectivity Gateway
End–user Device
1
2
3
4
5

Figure 8 The Hyacinth prototype with multiple channels [15]

The main goal of the prototype is to enable end-user devices to access the Internet or an enterprise internal network via the multi-channel WSN core which is composed by a group of wireless mesh router nodes and traffic aggregation devices. Each router node in the core is equipped with only two interfaces and each interface within a router node is bound to a distinct radio channel, considering the neighboring nodes within the signal interference range and the total number of channels initially allocated in the core. Any two router nodes within communication range should share at least one channel intended for direct communication with each other in an attempt to forward the data traffic originated from end-user devices to the wired network through the router gateway nodes. To achieve this goal, a fully distributed channel assignment algorithm along with a multiple spanning tree-based routing algorithm was adopted in the prototype to support dynamic traffic adjustment for the purpose of load balance.

It is evident that the prototype is partially scenario-specific with the algorithm employed above in terms of the direction of the data flows in the network, which prevents it from being extended to more general cases. Besides, the prototype only supports a static routing configuration in simulation scripts and forces all nodes involved to be generated with the same number of interfaces, irrespective of whether they are used in execution.

4.2.3 Dynamic Interface Extension to NS-2

The approach implemented in [16] to a great extent refers to the idea of interface construction within a mobile node proposed in [15], as illustrated in Figure 9.

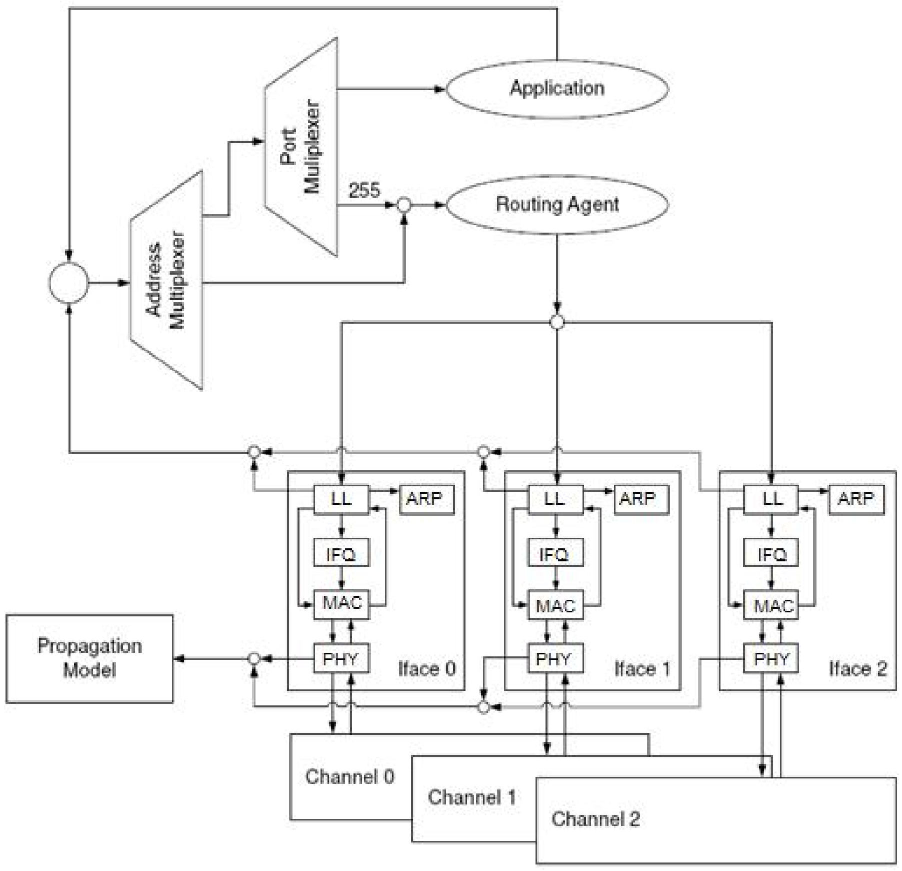


Figure 9 Dynamic interface extension intended for a mobile node [16]

Specifically, the authors designed and implemented an architecture for mobile nodes that is featured with a comparatively flexible configuration through simulation scripts, as compared to the approach in [15]. All nodes with a different amount of interfaces could co-exist in the same scenarios for communication. Meanwhile, each interface of a mobile node is assigned a single channel object, in which case packets passed down from the upper layer could be transferred respectively over multiple shared channels among nodes. In addition, all interfaces within a mobile node are attached to a dynamic routing protocol for the purpose of united packet forwarding with the support of the underlying interface index calculated from the corresponding MAC address.

To sum up, the approach only focuses on the scenario of multiple channels within an IEEE 802.11 network in the sense that a common radio propagation model is shared among all PHY layer objects when transmitting packets. Beyond that, it does not address the issue of address mapping from a node to multiple MAC objects attached to the node intended for the identification of packet addresses, since packets are transferred at the lower layer with the source and destination address distinct from the upper layer.

4.2.4 MW-Node

A Module-Based Wireless Node (MW-Node) is an innovative layout of a mobile node implemented by the authors in [17][18] to support multiple interfaces through reorganizing the existing components responsible for the functionalities of mobile node, as shown in Figure 10.

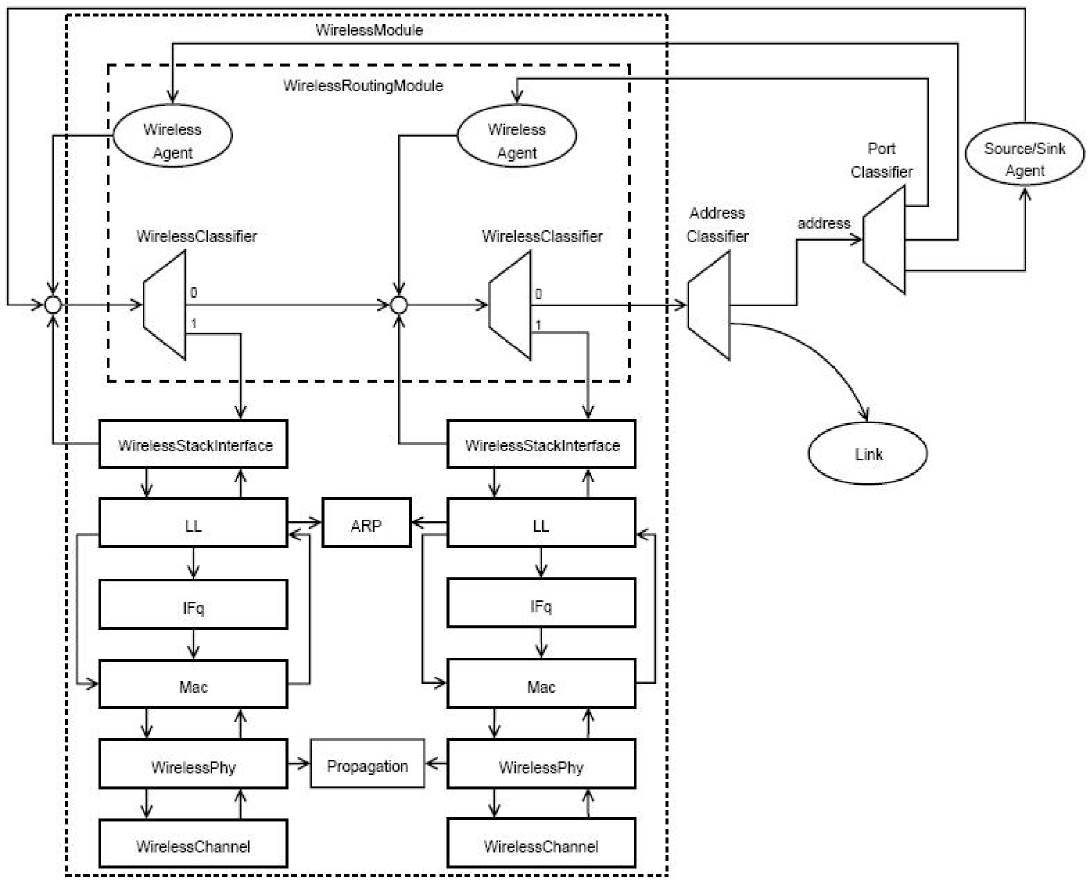


Figure 10 Schematic design of a MW-Node [17]

One of the major features in a MW-Node is that a routing protocol object originally linked to mobile nodes is replaced with a wireless routing module that functionally subdivides the routing protocol into two separate modules: a wireless agent and a wireless classifier. The wireless agent takes responsibility for generating packets associated with data routing and tagging them with the corresponding interfaces connected to it, whereas the wireless classifier is primarily used for forwarding the packet from the wireless agent or a source agent within mobile nodes to destination nodes. In addition to that, it is the wireless stack interface that tags incoming packets with the current interface identifier before handing them over to the wireless routing module.

Meanwhile, the author assumed that some of mobile nodes might play multiple roles at the same time in a wireless network. Therefore, a wireless agent along with a wireless classifier is created and interconnected thereafter in the wireless routing module so as to support multiple routing protocols within a mobile node when another interface is attached to the mobile node. As a consequence, a routing information base is shared among multiple routing protocols, which enables the wireless routing module to determinate over which interface to send packets originated from the upper layer or received from one of the underlying interfaces.

The main drawback of the architecture is that it is mostly oriented towards emerging routing protocols which could be built upon the subdivision mechanism at the sacrifice of backward compatibility with existing routing protocols in NS-2. In addition, interfaces of a mobile node fail to remain entirely independent from each other since they still share an ARP object as well as a radio propagation model at the lower level.

4.2.5 MIRACLE

The authors in [19][20] presented a Multi-InteRfAce Cross Layer Extension (MIRACLE) to NS-2, which serves as a generalized modular framework aiming at providing well-defined software interfaces for new features in NS-2 so as to facilitate simulations in the environment of multiple radio technologies. Specifically, the whole architecture is featured with modularization at each layer and cross-layer communication due to constructional design and functionalities involved, as shown in Figure 11.

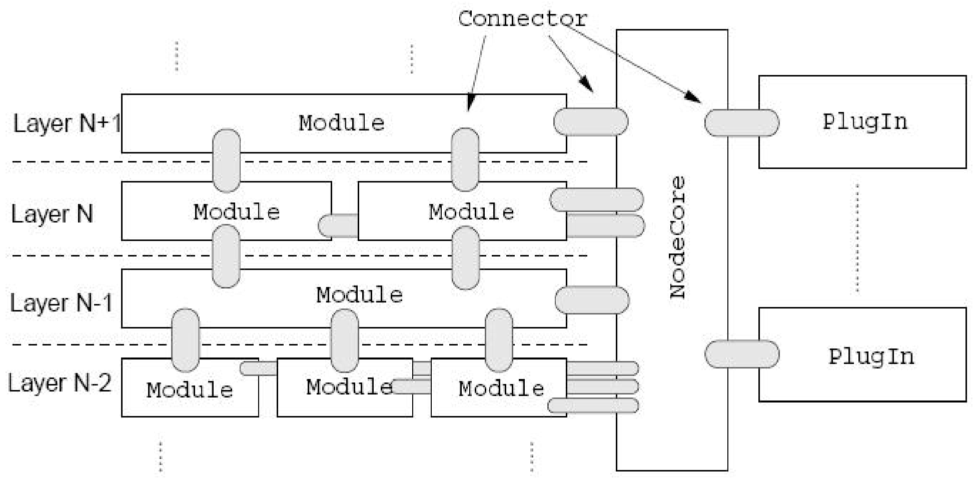


Figure 11 The general modularized architecture within the MIRACLE framework [19]

Each module in the architecture is an independent software container encapsulating a functional entity or a protocol object that includes application, transport protocol, routing layer, MAC, PHY, and so forth. Multiple modules with identical functionalities can share the same protocol layer and establish multiple protocol stacks independent from one anther by linking to their upstream and downstream modules with Connector objects.

Meanwhile, a unique structure called NodeCore placed at multiple layers acts as a coordinator and a manager that enables inter-module communication and provides common functionalities for all modules, whereas each PlugIn objects, attaching to the NodeCore holds cross-layer algorithms that are employed in message exchanges among modules. Consequently, a module is capable of communicating with any other module by sending messages through a Connector object with the support of the NodeCore and the PlugIn objects. With the cross-layer communication mechanism, the authors also developed a module called MIRACLE Physical Layer Module (MPhy) that provides a generic basis of the implementation of different wireless technologies oriented toward signal disturbance by introducing the calculation of Signal to Interference plus Noise Ratio (SINR) when receiving packets.

To reuse the existing protocol implementations in NS-2, the parameter interfaces of each protocol class and the mechanism of packet transferring within the protocol class have to be fundamentally adjusted to suit the demands of modules specified in MIRACLE, which seems to be a burdensome and time-consuming tasks to researchers from the perspective of backward compatibility. Meanwhile, the authors failed to cover the issue of whether or not the cross-layer object communication has a negative impact upon the normal packet delivery in terms of delay. In addition, the whole architecture and the corresponding architecture is scenario-specific to some degree in dealing with the issues of routing among multiple interfaces as well as how to utilize the underlying channel objects in a detached way, since the whole architecture was well-tuned by the authors to support their own Ambient Network, combining it with Universal Mobile Telecommunications System (UMTS) and IEEE 802.11 in the context of beyond 3G networks.

4.2.6 Conclusion

Undoubtedly, there is no alternative that is appropriate to address generic issues that arises in the context of MIMC among multiple networks. Existing solutions to the issues of MIMC are project-specific in the sense that the architectural design of nodes and the majority of procedures in handling packets are mostly tailored to a special scenario such as multiple interfaces with a single radio technology, a mixture of wireless and wired network, etc. In terms of software implementation, technical tradeoffs always exist between the architecture of mobile node and the internal mechanism of protocol objects during data transmission. Even so, a wide range of methodologies with regard to the node construction as well as packet forwarding were explored in these solutions which lay the foundation for the implementation of our simulation model.

4.3 Solution to Nodes with Multiple Interfaces/Channels

The solution to our project is partially inspired from the idea of dynamic configuration in [16] and the modularity of node construction in [19]. Unlike these solutions mentioned above, we focus on the scenario of multiple channels, each of which belongs to a single network. Meanwhile, packets issued by one mobile node in one of these networks are required to be transmitted to any destination node over other networks based on the scenarios specific to our implementation. Thus, the construction of a mobile node is the first step to be taken to deal with the issues of MIMC in our project. The implementation mechanism of protocol layer objects should be modified according to the established framework so as to meet our demands in smart home networking simulations (discussed in the following chapters).

4.3.1 Architecture of Mobile Node with Multiple Interfaces

To clarify the layering and modularized structure of mobile nodes in our solution, a general architecture of a mobile node is shown as follows:

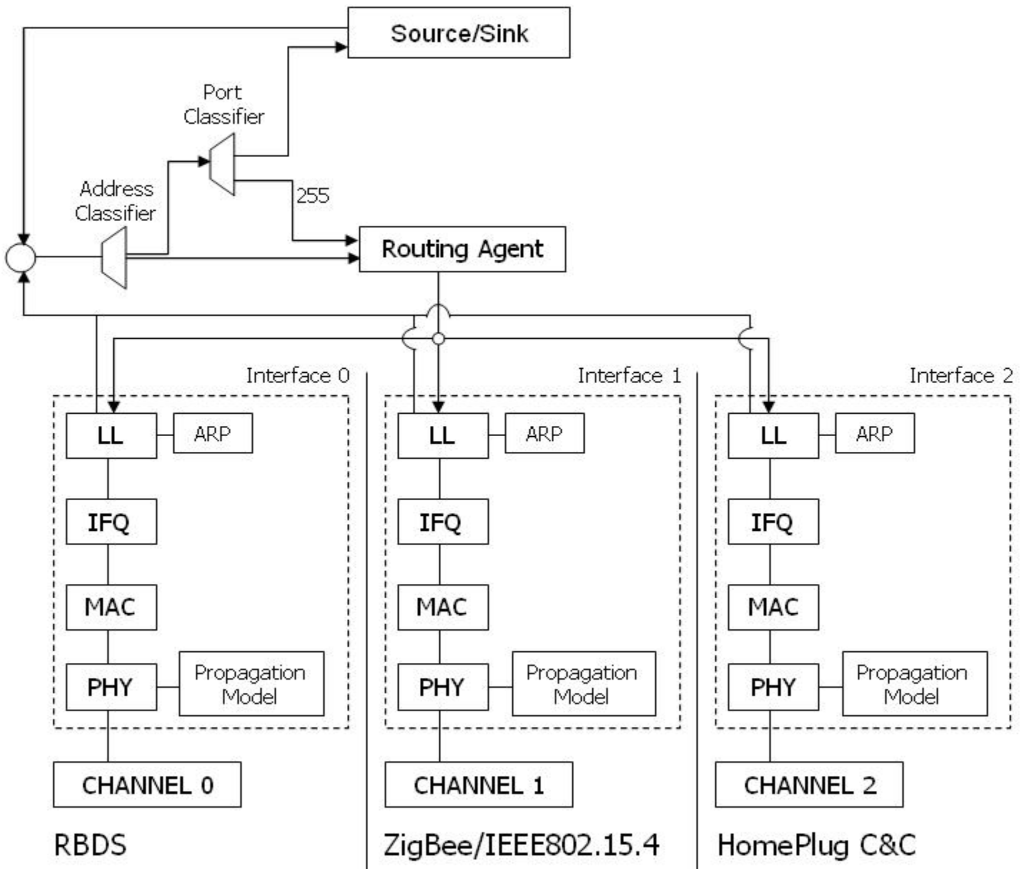


Figure 12 General architecture of mobile node with multiple interfaces/channels

In terms of the functionalities of each object in the architecture, there is no difference between our solution and the original framework of a mobile node existing in NS-2. However, the number of supported protocol stacks below the routing agent dynamically increases. Equipped with a unique index, each set of protocol stacks within a mobile node serves as an independent interface linked to an underlying wireless network for the purpose of data transmission. In order to do so, each physical layer object holds a single radio propagation model object configured in advance for its own use. Meanwhile, each channel object oriented towards one network in our simulation is independent from others with respect to communication-related parameters that were originally shared in NS-2 (discussed in Chapter 5). In addition to that, the static numbering allocation for channel existing in NS-2 is also kept so as to enable each channel object to distinguish itself from others, which entirely identifies each wireless network in simulations.

Currently, there are four types of mobile nodes involved in our project: one node with one interface used for RBDS message broadcasting (the RBDS message originator), one node with three interfaces (the smart home central controller) mainly responsible for RBDS message forwarding from the RBDS network to the other two networks in the smart home, a group of intermediate nodes with two interfaces acting as both routers with multiple paths and RBDS message recipients, and the remaining nodes with one interface active for data transmission, functioning as ordinary nodes in the ZigBee/IEEE 802.15.4 network or the HomePlug C&C network. To handle packets received from different networks for all types of nodes, the node entry point is concurrently linked to multiple interfaces available in the node so as to pass incoming packets up to the corresponding layer object. Besides, the routing agent has to make decision of which interface to employ for outgoing packets, depending upon the routing strategy as well as the mechanism of packet tagging (discussed in Chapter 6).

4.3.2 Parameter Binding of Energy Model to Multiple Interfaces

One of the properties of mobile nodes is the energy model that represents the energy level for the node. At the beginning of simulations, the mobile nodes in a network can be configured with an initial energy value and preset power levels for each packet to be received and transmitted. Therefore, the data transmission in different networks will decrease the energy level of mobile nodes, which in turn prevents nodes from handling any incoming or outgoing packets when their energy capacity is reduced to zero.

Originally, the calculation of power consumed for packet reception and transmission are conducted by only one PHY object with the energy level passed down from the given node object in NS-2, since it is assumed that only one network exists, as illustrated on the left side in Figure 13:

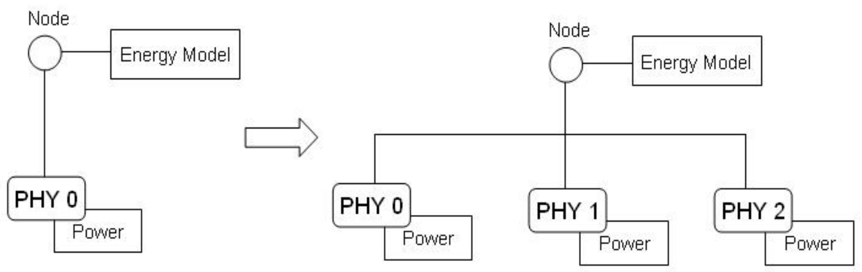


Figure 13 Modification of parameter binding for the energy model

Considering the fact that the setting of power levels at the physical object indexed 0 has already been done in NS-2, all the PHY layer objects within a mobile nodes are configured with the power levels respectively specified in scenario scripts through a group of united parameter setting methods in our implementation in such a way as to maintain backwards compatibility with the existing mechanism in NS-2. In this case, the energy level of a mobile node could be passed down to each PHY layer object for the calculation of power consumption when receiving or transmitting packets upon request.

4.3.3 Object Binding of Error Model to Multiple Interfaces

The error model supported by NS-2 is considered an independent object that simulates packet errors or loss over communication links that are produced by various methods of error calculation. After determining an incoming packet to be a corrupted one, the error object either sets the error flag of the packet or directly dumps it to a drop target according to the configuration in the scenario scripts. Given the direction of data transmission, an error-tagged packet is forwarded to the corresponding protocol layer object pointed to by the error object for the purpose of further handling, whereas a corrupted packet without marking the error flag is simply discarded by the drop target. Also, the error object enables error-free packets to go through without any intervention to the next protocol layer. Hence, we chose to discard the corrupted packets generated by a random variable at the receiver side so as to satisfy the requirement of reliability in simulation scenarios.

For a mobile node over wireless networks, an error model is inserted between the MAC layer and the PHY layer, as illustrated on the left side in Figure 14.

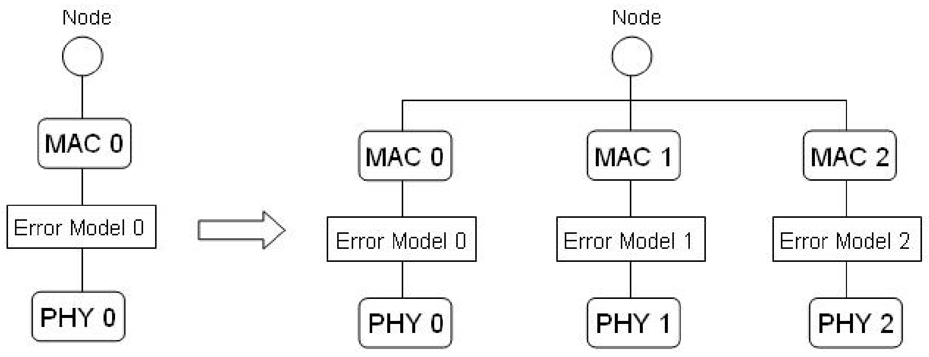


Figure 14 Modification of object binding for the error model

Similar to the energy model, the existing NS-2 simulation environment only provides support for one mobile node bound with one error model. To simulate the packet loss oriented towards each interface attached to one mobile node over multiple networks, each interface is equipped with one error object preset with an individual packet error rate in our implementation. As a consequence, each error object is capable of handling incoming packets from the corresponding interfaces connected to it separately, which enables mobile nodes to forward packets via multiple networks, depending upon their reliability during data transmission.

4.3.4 Logic of Connection to Multiple Interfaces

In NS-2 simulations, interpretation-based OTcl objects are generated step by step based on the scenario script, whereas compilation-based C++ objects are generated afterwards through the mechanism of static shadow mapping from the OTcl space to that of C++. To guarantee correct connections among layering objects in both interpreted and complied hierarchies, the construction of a mobile node should strictly follow the order of creating network layering objects for each interface within the node, as shown as follows:

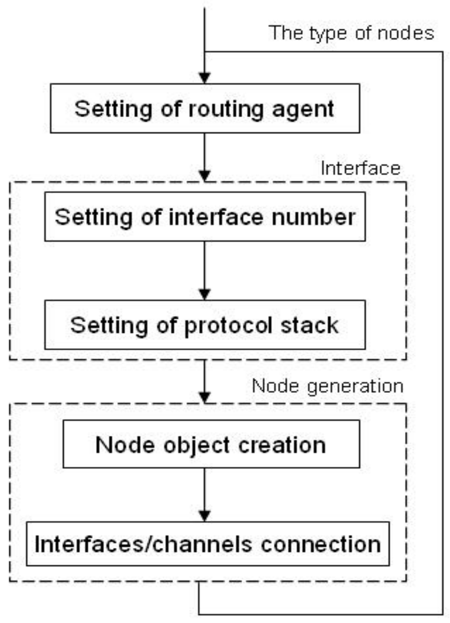


Figure 15 The flow chart of node construction in simulations

The figure above illustrates the general procedure of how to establish valid connections for each interface within a node for each type of nodes in our project. Meanwhile, the procedures are set once for each type of nodes in multiple networks before the nodes are generated. To be specific, the procedures are subdivided into three parts that cover the setting of the routing protocol type, the configuration of multiple interfaces, and the node generation.

First of all, only the routing protocol type is configured for the first node to be created in the scenario. In terms of multiple networks, the type of routing protocol varies with the functionality of nodes as configured in advance. Secondly, the number of interfaces needs to be initialized by a default value without specifying any layering object along with it, which allows the NS-2 simulator to generate a group of object arrays rather than individual objects, given the mechanism of dynamic type binding inherent in OTcl (the type of an variable are not determined until the variable is assigned a value with a type automatically recognized in OTcl). Each object array stores the same type of layering objects involved in multiple networks prior to node construction.

Subsequently, the number of interfaces and the type of underlying protocol objects in each interface are specified for the current type of nodes. Unlike the layering objects, each propagation model object belonging to one PHY layer object within a node must be generated in advance to avoid collision with the existing procedures of node construction in NS-2.

The last part for node construction is to instantiate node objects and to establish the connections among layering objects within the nodes from top to bottom with all configurations prepared in previous parts. Underlying network objects within a node are created for each interface after the generation of the node object. Besides, all interfaces along with the corresponding communication channel object (generated before any node is created) are linked to the node one by one through a loop. Additionally, the order of different types of interfaces for each type of node is statically assigned specific to our design in later cooperation with the complied objects in transferring packets among multiple interfaces.

To sum up, the type of network objects are specified in Tcl scripts, whereas the logical correlation among these objects based on the framework of node is established in the scope of OTcl. Besides, the compiled network objects must adhere to the new correlation in packet handling and packet forwarding across multiple networks during simulations. Hence, it is necessary that the modifications to each network object involved be explained from the perspective of operational mechanism and implementation logic in the following chapters.

5 Communication Channel in NS-2

5.1 Operational Mechanism of an NS-2 Channel

A Channel object acts as a logic abstract in NS-2 to simulate the realistic transmission of packets over the shared physical medium. To achieve it, the Channel object must calculate the total number of neighbor nodes within the reach of the transmission signal as well as the communication delay involved in the propagation model before scheduling the packets to destination nodes with the help of the scheduler. Thus, the primary function of the Channel object is to manage all nodes attached to the same channel, as illustrated in Figure 16:

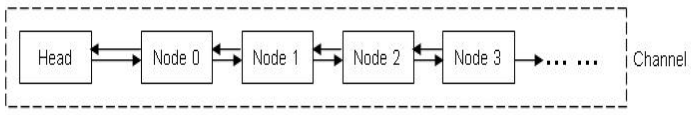


Figure 16 Organization of nodes over a single channel

Each node object within a shared channel is connected the others through a bi-direction linked list held by the Channel object when it is generated with the same channel in the program space of an NS-2 simulation. In this way, the Channel object is capable of pinpointing any destination node in its list that is eligible to receive packets as long as these nodes are kept unchanged in the list (nodes could be added and removed in special cases).

NS-2 was originally not designed to support multiple networks coexisting in the same scenario for simulations. As a consequence, the relationship between nodes and a linked list has to be redefined for the purpose of the situation of multiple interfaces/channels. Considering the fact that each channel in the scenario is only occupied by a single network, a matrix of linked lists is constructed for nodes and multiple networks as illustrated in Figure 17.

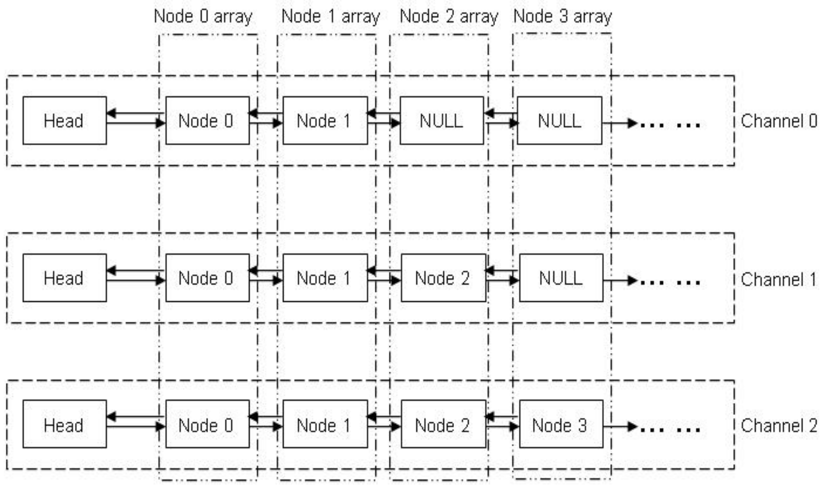


Figure 17 Array management of each node across multiple channels

The figure above gives an example of nodes with three interfaces/channels. In our implementation, channel 0 is assigned to the RBDS network, whereas channel 1 and channel 2 are assigned to the ZigBee/IEEE 802.15.4 network and the HomePlug C&C network respectively. Moreover, a bi-directional pointer array intended for multiple channels is created for each node so as to connect to other nodes over the same channel in a horizontal manner. In such way, each node is only located over the same channel when transmitting packets in the same network without any impact upon data communication over other channels in the sense that nodes are vertically attached to linked lists that belong to distinct channels.

5.2 Decoupling of Configuration for Multiple Channels

The authors of the NS-2 implementation assumed that all nodes in simulations should share the same parameters for data transmission associated with the physical layer and the corresponding propagation model in that they communicate with each other within a single network. Under this assumption, all variables related to the range of the transmission signal are statically declared in the Channel class so as to be employed among all nodes in the network (calculation is executed only once during the whole simulation by the very first node that transmits packets), as illustrated on the left side in Figure 18.

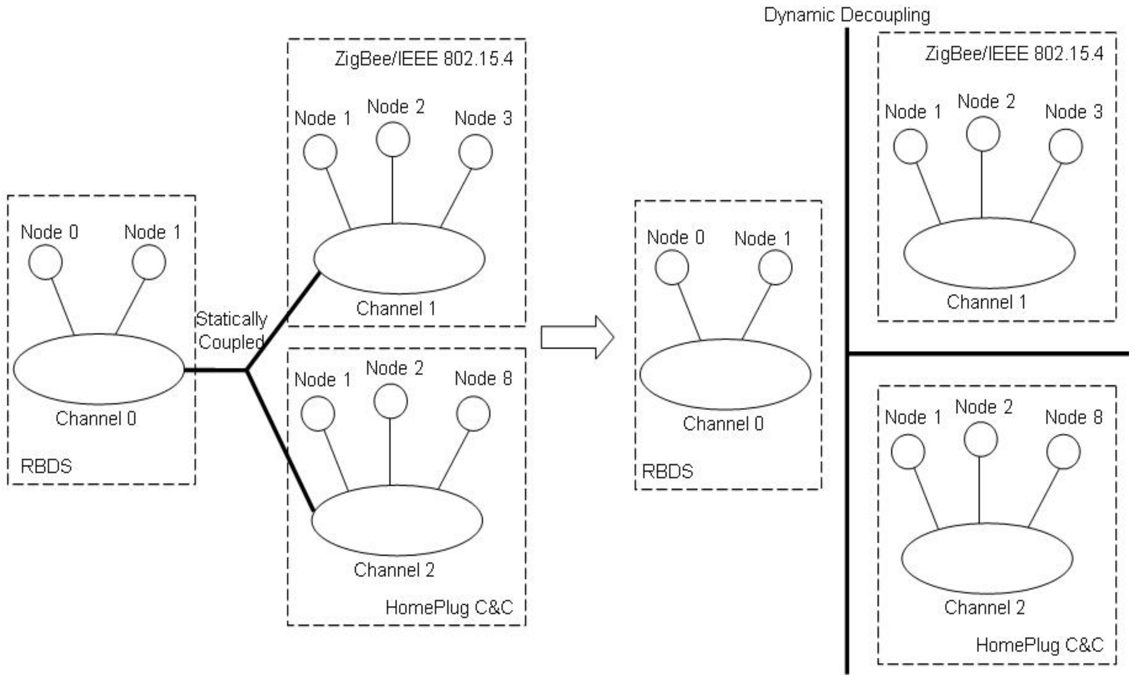


Figure 18 Decoupling of parameter association among channels

Nevertheless, this is not the case in our simulation model, in which different networks hold a stand-alone channel with a physical layer different from other networks and the corresponding propagation model configured. Besides, the propagation model intended for each network distinguishes itself from each other in terms of parameter configuration even though these networks are configured with the same propagation model. To address this issue, the static property of those variables and the static assignment to them have to be replaced with the dynamic declaration and initialization in the Channel class for each channel object generated in the NS-2 environment (on the right side in Figure 18). Each channel object holds its own parameters relevant to data transmission that has no connection to other networks in such a way as to avoid issues of packet loss and abnormal transmission in each network during simulations.

5.3 Packet Delivery among Multiple Channels

Originally, all nodes share the same channel for data transmission in the case of one network with a single channel in the NS-2 environment. Packets could be delivered to destination nodes by default after the calculation of transmission range and propagation delay. For the situation of multiple channels occupied by multiple networks, there are two steps to be taken so as to enable the PHY layer of destination nodes to identify the right packet when transmitting a packet. First of all, all neighboring nodes affected by the transmission are filtered out, since all nodes are organized in each channel respectively. Secondly, an address pointer to the right PHY layer object of each node that is directly connected to the current channel object and the value of propagation delay are placed as key fields in the packet header before it is inserted as an event into the scheduling queue exclusively manipulated by the scheduler object, as shown as follows:

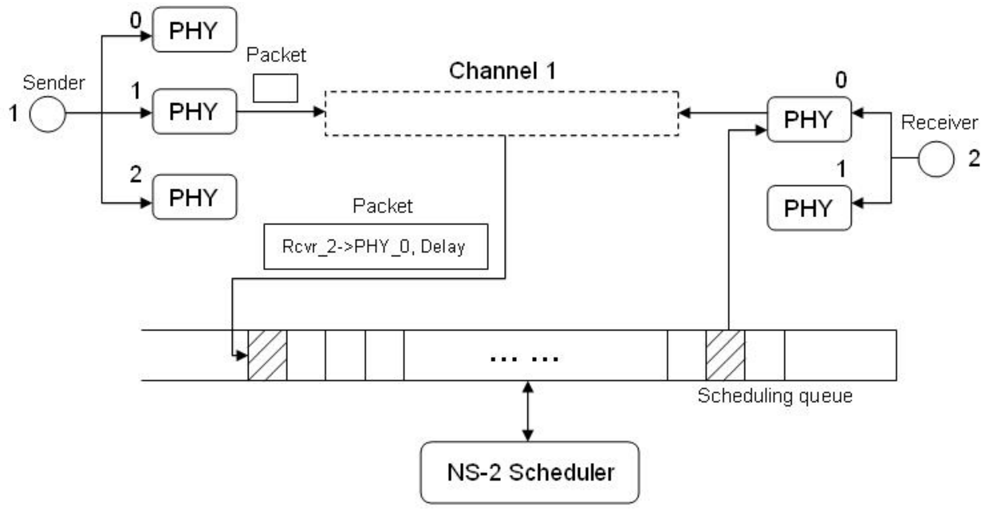


Figure 19 Data transmission on the basis of object identification and packet scheduling

After the interval specified by the propagation delay, the packet is picked up from the queue and dispatched by the scheduler object to the PHY layer of the destination node through the callback mechanism of C++ objects in that the address of any PHY layer object is unique in the execution space of a NS-2 program. Besides, it is the routing layer object that takes responsibility for forwarding packets among multiple networks, since each PHY layer object of nodes passes packets up to upper layers for further handling.

Normally, mutual interference of signals among channels exists in on-site experiments executed in a realistic environment when more than one channel of a node is simultaneously involved in data transmission. To take that factor into account in NS-2, a mechanism of inter-object message communication is supposed to be implemented in all

PHY layer objects within one mobile node in the sense that each PHY layer object is capable of measuring and calculating the influence of interference by communicating with other PHY layer objects when transmitting or receiving packets. However, such an approach is technically sophisticated in terms of the existing mechanism and the architecture of software inherent in NS-2 since it deals with the construction of nodes and the reestablishment of the existing procedures of packet scheduling. Moreover, it could also fundamentally alter generic behaviors in simulations, which is far beyond our focus in this project.

Therefore, we exclude this factor from our implementation so as to simplify the networking model. In other words, there is no way for data transmission of any kind in one network to disturb other networks in our solution on the basis of the underlying packet delivery mechanism, the uniqueness of the address of each PHY layer object as well as the independence of each channel object (simulations of multiple networks with a Flooding protocol configured are adopted to demonstrate the mechanism).

6 MAC/PHY Layer

6.1 Interface Numbering and Packet Tagging

Nodes are categorized into four types intended for multiple networks in our model that includes a RBDS message sender with one interface, a central controller with three interfaces, intermediate nodes with two interfaces, and the remaining nodes with one interface active in use. Each kind of interface within a node is attached to one channel object, as illustrated in Figure 20.

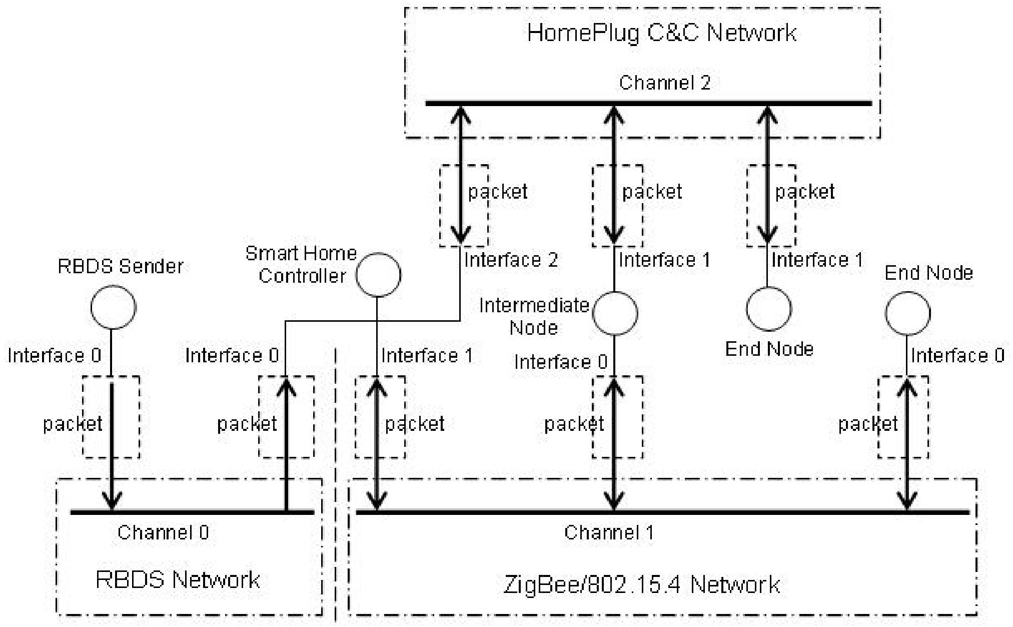


Figure 20 Node classification and packet tagging across multiple networks

In NS-2, a protocol object within a node has little knowledge of the operational status of other protocol objects due to the independence of objects at each layer, which seems quite common in ordinary simulations. Nevertheless, this is not the case in the scenario with multiple networks involved. For one thing, any node engaged must be notified in advance of the total number of interfaces currently connected to it for the purpose of proper execution; for another, the routing layer object of nodes has to make routing decision for outgoing packets on the basis of packet type as well as the identifier of interfaces corresponding to a specific network.

Therefore, there are two steps that are taken to guarantee that packets are transferred from the RBDS network to the mixture of networks in a smart home rather than to be delivered in the reverse direction. In the first place, each interface configured in a node is explicitly specified with an integer identifier starting from zero in the scenario scripts at the beginning of simulation. In the second place, all incoming packets passed through the current interface are tagged with the corresponding interface identifier before handed over to the upper layer. In such way, packets could be forwarded through the right interfaces to destination nodes.

6.2 RBDS

To model packet transmission in the RBDS network, we adopt the RBDS MAC/PHY protocol stack implemented in [10]. The radio propagation model at the RBDS PHY layer is a combination of the Shadowing model and the Ricean/Rayleigh fading model, which concurrently simulates the large scale fading effect of a wireless channel and burst errors on the basis of the time correlation of the received signal in packets. The RBDS MAC layer employs the TDMA-based scheme modified in the existing NS-2 implementation to allot all the time slots to a RBDS message originator in our multi-interface model for data broadcasting in such a way as to mimic the operation of FM radios. Another node acting as a central controller in a smart home is configured to stay tuned to the broadcast from the RBDS message originator in each time slot. Meanwhile, the destination address of each packet issued by the RBDS message originator is set to be the broadcasting address before transmission.

By default, the RBDS message originator, indexed 0, mainly takes responsibility for transmitting broadcast packets through the underlying interface 0 connected to the node in our simulation model. Without sending time slots allocated, the central controller, indexed 1, simply receives all packets over the RDBS network through the interface with the same index and routes them to destination nodes in a smart home through the remaining interfaces attached to the same node.

6.3 ZigBee/IEEE 802.15.4

The IEEE 802.15.4 MAC/PHY protocol stack already included in the core release of NS-2 implementation is adopted in our project to fit in the corresponding part of underlying interfaces attached to a mobile node. Considering the fact that the IEEE 802.15.4 MAC/PHY protocol implementation serves as an ideal model intended for software simulation, part of its operational mechanisms remains distinct from that of off-the-shelf ZigBee/IEEE 802.15.4-based devices in realistic environments due to the intrinsic characteristics of NS-2. Also, the organization of nodes in a ZigBee/IEEE 802.15.4 network and the choice of the mode in packet transmission have to be taken into account in terms of the node construction and the setting of simulation scenarios.

6.3.1 Characteristics of Software Implementation

First of all, the node type and the corresponding functionalities partially conform to the ZigBee/IEEE 802.15.4 technical specifications. In the normal case, devices configured in a network are classified into two types: Full Function Device (FFD) and Reduced Function Device (RFD) [21]. A FFD could be configured to operate in two modes: a Personal Area Network (PAN) coordinator or an ordinary coordinator. In terms of functionalities, a PAN coordinator takes responsibility for the establishment and management of the whole network, whereas ordinary coordinators serve as sub-roots or routers of the network that connect RFDs to the network and for the purpose of data forwarding. A RFD represents a resource-restrained device that only communicates with the ordinary coordinator connected to it. Contrary to the specification and real devices, there is no strict classification that identifies the type of node in NS-2 except the functionality of node association (explained in next section). In such case, a RFD is capable of transmitting or routing packets to any node within its signal range instead of transferring them to the sub-root coordinator regardless of its type. Therefore, the data communication among nodes is executed in a peer-to-peer way in our implementation.

Secondly, there is no direct support for sleep mode in the activities of nodes during simulations. The ZigBee/IEEE 802.15.4 specification has been originally designed to minimize the active energy usage of standard-compliant devices. To keep nodes at the lower level of power consumption during a long-term operation, each node in a network should remain dormant during the majority of operational time, except that it is re-triggered from dormancy to respond to periodical events [22]. To idealize the execution, the NS-2 simulator treats the power consumption in idle mode the same as in sleep mode on the basis of the energy model configured to nodes. Therefore, the power levels of both idle mode and sleep mode that are assumed to be extremely low in simulations are set to identical values for simplicity.

In addition, the choice of the logic number of a communication channel has nothing to do with the static identifier assigned to the underlying channel object. According to the specification, a total of 27 channels starting from 0 to 26 are available across three frequency bands in which the ZigBee/IEEE 802.15.4 protocol stack can operate (250 kbps in the 2.4 GHz band, 40 kbps in the 915 MHz band, and 20 kbps in the 868 MHz band) [21]. In such case, channel 0 is assigned to the 868 MHz band; 10 channels numbered from 1 to 10 are assigned to the 915 MHz band; and 16 channels numbered from 11 to 26 are assigned to the 2.4 GHz band. Given the predetermined frequency band, the PHY layer is capable of tuning its transceiver to a certain channel within the numbering scope upon receipt of packets from the upper layer. From the perspective of simulation implementation, the logic number of a channel is calculated by the PHY layer object with a preconfigured frequency band in an attempt to tag the wireless link among nodes in data communication, whereas the channel identifier, statically assigned to the channel object, is uniquely recognized and handled by the simulator during the run-time scheduling of NS-2. Hence, there is no mapping of any kind between the two types of identifier for the same channel object in that they are employed for different purposes.

6.3.2 Network Topology and Node Association

In terms of networking architecture, a ZigBee/IEEE 802.15.4 network could operate in either the star topology or the peer-to-peer topology [21]. In a network with a star topology, the communication is established in a centralized way, in which case devices only communicates with the PAN coordinator in the network. Different from the star topology, any device in a network with a peer-to-peer topology is capable of transmitting packets to any other device within its radio range. Meanwhile, the peer-to-peer topology enables nodes to transfer packets through multiple hops to destinations, depending upon the routing strategy at the upper layer. As a matter of fact, a single cluster tree network intended for our simulation scenarios is a typical case of the peer-to-peer topology. Under such circumstances, a FFD or RFD joins the network as a leaf device at the end of a branch of the cluster tree by connecting itself to a FFD equipped with the capacity of association.

Device association acts as a key step to form a cluster tree topology which is a special network structure that could be exploited for the routing purpose if necessary. To conduct data transmission, any node must join a ZigBee/IEEE 802.15.4 network through device association procedures so as to identify itself as a member in the network, as illustrated in Figure 21.

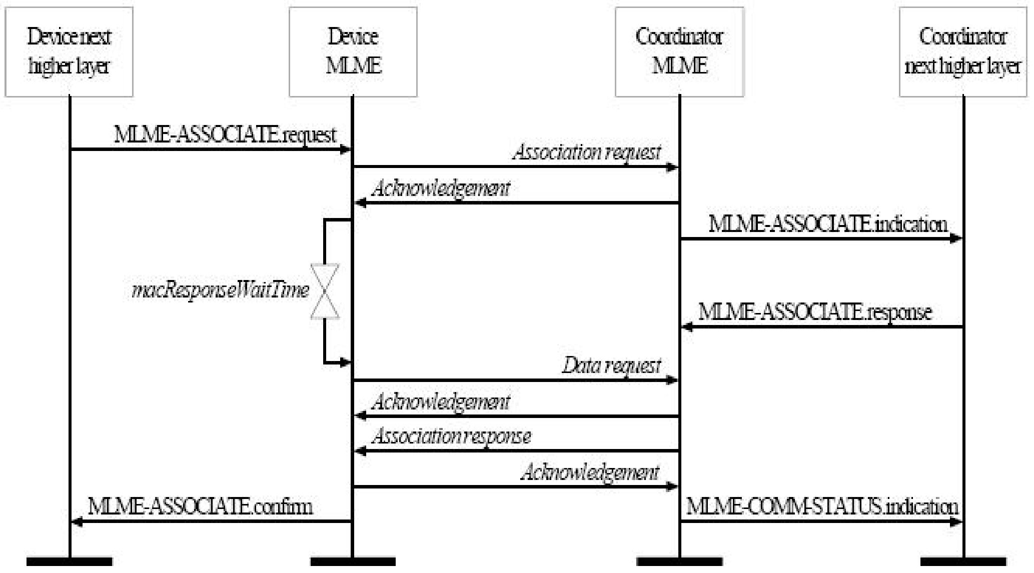


Figure 21 Association sequences of device in a network [21]

First of all, the first node intended for a ZigBee/IEEE 802.15.4 network is set to be the PAN coordinator through scenario scripts. The PAN coordinator establishes the network by choosing an unused PAN identifier unique in the scope of simulations. Following that, any other node is able to join the network for data communication with its node identifier as a fixed address in simulations.

A normal device association process begins with a communication channel scanning initiated by an unassociated device in an attempt to locate the existing network and the surrounding coordinators (intermediate nodes or the central controller in our project) that are configured to receive association requests from devices. Subsequently, the device issues an association request with an acknowledge requirement to the coordinator. Upon reception of the request, the coordinator has to make the decision of whether to allow the requesting device to connect to the network through it based on its capacity and resources available for association. Whatever the decision, it is up to the device that issues another packet requesting the feedback from the coordinator after macResponseWaitTime symbols in the indirect communication.

6.3.3 Packet Transmission

From the perspective of the MAC layer, the mode of transmitting packets could be categorized into indirect transmission, Guaranteed Time Slot (GTS) transmission, and direct transmission [21]. Indirect transmission is mainly characterized with the one-way communication originated from a coordinator to a device directly attached to it, in which case a device could extract pending packets held in a transaction list by its coordinator by tracking notification frames from the coordinator in a beacon mode or by polling the coordinator in a non-beacon mode. GTS transmission is only used for data transfer between a PAN coordinator and a device connected to the network. A GTS is only allocated by the PAN coordinator to an associated device upon request. A PAN coordinator is capable of allocating up to seven GTSs simultaneously, each of which is bound with the address of an associated device and a fixed direction for data transfer. Direct transmission applies to data commutation of all kinds regardless of the topology of the network and the type of nodes engaged.

It is evident that both indirect transmission and GTS transmission are unsuitable for generic peer-based packet transmission in a ZigBee/IEEE 802.15.4 network, considering the limitations inherent in their implementation mechanism. Meanwhile, the authors of the ZigBee/IEEE 802.15.4 MAC/PHY implementation demonstrated in their experiments that the hop delay in indirect transmission and GTS transmission is much longer than in direct transmission and in turn affects the packet delivery ratio at upper layer [26]. Also, their findings showed that GTS transmission is costly in terms of utilization. Hence, indirect transmission is only employed in association procedures in order to establish a network, while direct transmission is adopted for generic data forwarding in our simulation model, as illustrated in Figure 22.

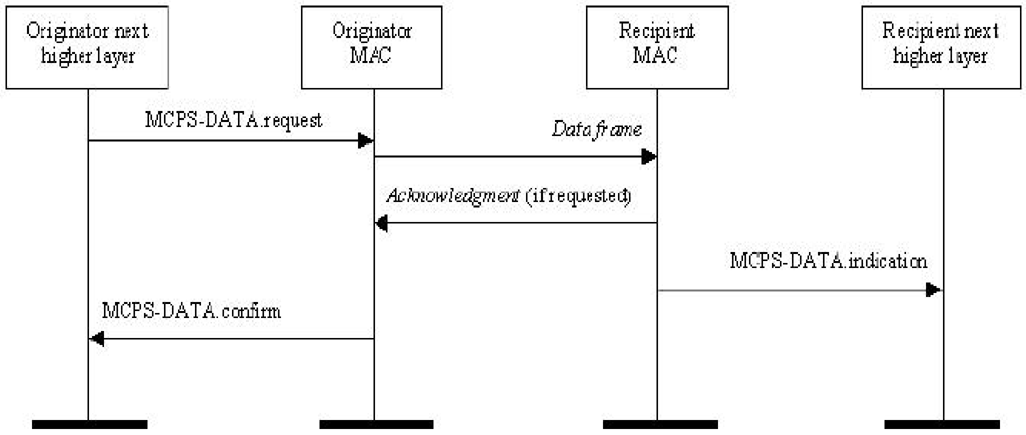


Figure 22 Simplified diagram of direct transmission among nodes in a network [21]

Initially, the MAC layer of an originator straightforwardly issues outgoing packets to destinations once it receives the data request along with the packets passed down from the upper layer. Upon reception of each data packet, the MAC layer of a recipient responds to the originator with an acknowledgement packet and notifies its upper layer of the incoming packet for handling. The upper layer of the originator ultimately receives confirmation messages representing successful deliveries from the underlying MAC layer. Considering that the whole process of data transmission is conducted through the shared wireless medium, it is highly likely that a packet of any type could be discarded due to packet corruption or the collision of access to the channel with other packets after retransmissions based on the CSMA/CA mechanism. Therefore, a notification of failure to transmission is eventually passed up to the upper layer for the purpose of decision-making of other outgoing packets.

6.4 HomePlug C&C

Initially, a power line was designed to transmit 50 or 60Hz signals instead of high frequency signals. Similar to a wireless channel, a power line channel suffers from a range of signal attenuations, including noise originated by the operation of neighboring electrical devices, fading, multi-path effect and interference caused by the limitation inherent in channels [23]. Based on the commonality between two types of communication medium, the authors in [24] demonstrated the feasibility to model the power line communication in the NS-2 simulation by attaching a Korea Standard power line MAC (KS-MAC) protocol component to the built-in architecture of a mobile node, as illustrated in Figure 23.

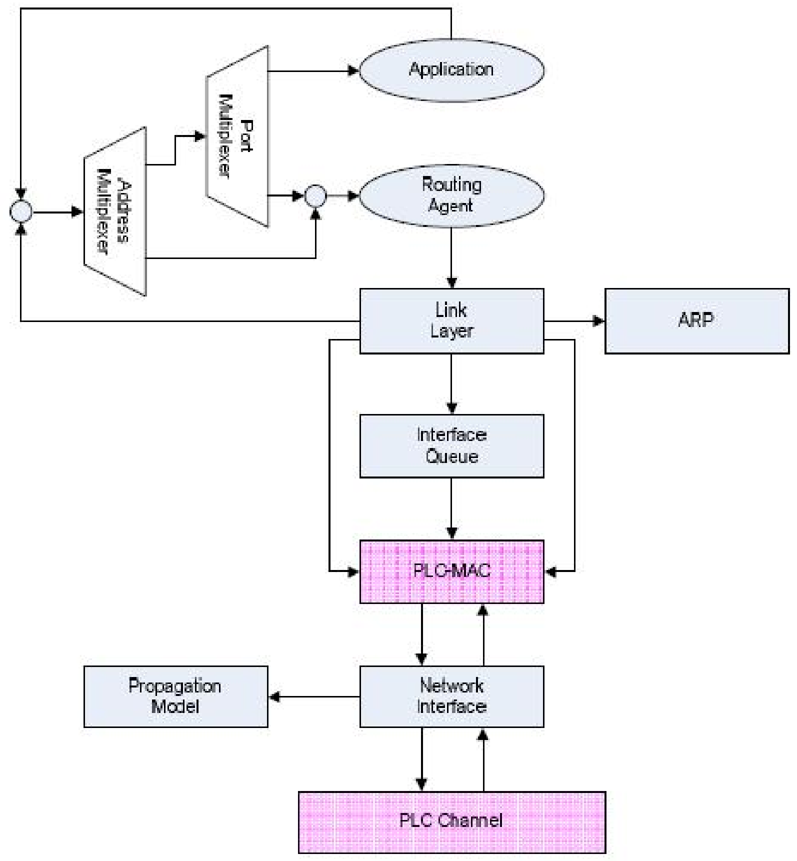


Figure 23 The architecture of a KS-MAC based PLC-node in the NS-2 extension [24]

In this architecture, the KS-MAC layer object mostly takes control of the access to the share medium with the channel contention mechanism in addition to such basic functionalities as transmission and reception, segmentation and reassembly, error control, etc. To accomplish the goal, the channel estimation module in the KS-MAC layer object independently collects the current status of the channel so as to calculate an adaptive contention window.

Inspired by the characteristics of a power line channel as well as the approach to construct a PLC node in NS-2, we chose to clone the IEEE 802.11 WLAN protocol stack along with the corresponding communication channel from the existing NS-2 implementation with a date rate reduced to a low value (25Kbps) as compared to that of ZigBee/IEEE 802.15.4 (250Kbps at 2.4GHz). This creates a technical approximation of the HomePlug C&C MAC/PHY (with a data rate of 7.5Kbps) since there is no detailed technical specification and software implementation publicly available for reference and adoption. Further bandwidth reduction is restricted mainly by the operational mechanism of NS-2 simulation environment as well as the complexity of event scheduling in the cloned implementation. Meanwhile, the newly cloned channel class and the existing radio propagation models are tailored to this substitution in terms of the computation of propagation delay in packet transmission.

7 Routing Layer

The routing layer in our simulation model mainly deals with issues including packet identification and data forwarding from the RBDS network to the combination of wired and wireless networks in a smart home, the way of routing data to all nodes configured to receive RBDS messages, and the establishment of routes for nodes across multiple networks in a house. The following sections in this chapter are organized to discuss the corresponding operational mechanism of each issue from the perspective of our model implementation.

7.1 Packet Transmission from RBDS to Smart Homes

The simulation model in our project consists of two types of networks: an outdoor network intended for long distance wireless transmission and multiple indoor networks interconnected through intermediate nodes in a smart home to enable short range packet forwarding. Normally, only the RBDS message sender preset in our model periodically broadcasts packets over a RBDS network to a smart home central controller within its signal range. Outgoing RBDS packets are initialized with the corresponding packet type at the RBDS originator side and tagged with the RBDS interface identifier at the recipient side during transmission. Upon reception of incoming RBDS packets at the routing layer, the routing object of the central controller handle takes responsibility for forwarding packets to destinations based upon a given routing strategy, as illustrated in Figure 24.

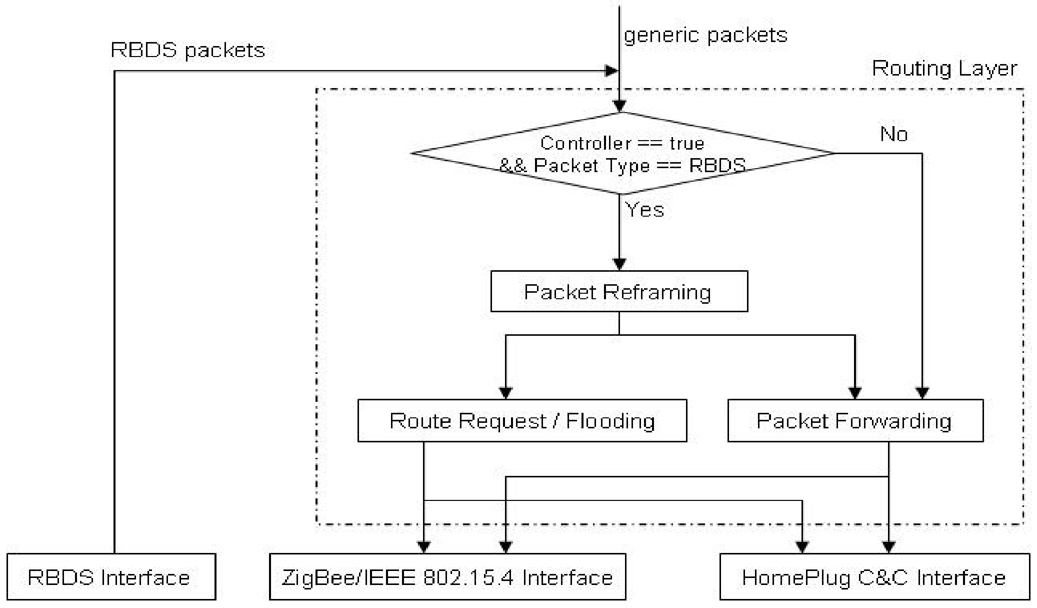


Figure 24 A combined flowchart of RBDS packet forwarding through the central controller

In the first place, the routing object of the central controller distinguishes different packets to be transmitted according to the current node type and the packet type along with a tagged interface identifier in that all nodes in a smart home are configured with a common routing protocol intended for packet forwarding over multiple networks. Incoming packets from the RBDS interface are intercepted by the routing object for further handling, whereas generic packets from the upper layer or other networks are directly passed down to the corresponding interface with the established route entry to destinations in the routing table.

In the second place, the routing object addresses issues in packet re-encapsulation and packet forwarding for incoming RBDS packets based on the chosen routing protocol (Flooding, AODV and ZigBee routing protocol are employed in our simulation model for the purpose of performance comparison). To be specific, outgoing RBDS packets are reframed with distinct source and destination address as well as other route-related parameters specific to the configured routing protocol, as compared to the original ones. In other words, the source address of packets is set to the address of the central controller while the destination address of packets is set with each node configured to receive RBDS packets in scenario scripts. Subsequently, the reframed RBDS packets are delivered to destinations by the routing object with the support of the routing protocol configured in simulations. Flooding-based packets could be directly handed over to all other interfaces except the RBDS interface. For AODV and ZigBee routing protocol, the routing object must determine whether to issue route request (RREQ) packets to destination nodes by examining whether a route entry with outgoing interface identifier exists in the routing table.

To sum up, all nodes configured in a smart home, including the central controller, intermediate nodes and generic nodes, are implemented to follow the same logic of handling outgoing packets. In terms of functionalities for AODV and ZigBee routing based nodes, it is evident that only the central controller is in charge of forwarding RBDS packets from the RBDS network to home networks, while the remaining nodes in a house handle incoming RBDS packets or issues status updates to the central controller through registered interface indexes that identifies the corresponding networks in their own routing entries.

7.2 Data Forwarding in a Smart Home

According to the ZigBee specification [25], a ZigBee/IEEE 802.15.4-based device should be equipped with a ZigBee routing protocol when transmitting packets after device association. As a variation of AODV routing protocol, the current implementation of the ZigBee routing protocol enables ZigBee/IEEE 802.15.4-based devices to deliver packets to destinations with a minimal link cost calculated from the corresponding packet delivery probability over the underlying communication link, whereas an AODV-based device employs the shortest hop count in packet forwarding. Therefore, both ZigBee and AODV routing protocol are modified to accommodate themselves to the simulation environment of multiple networks for the purpose of a close comparison between them. Also, an improved Flooding protocol is adopted in our project as another alternative to a redundant and robust data transmission among multiple networks.

7.2.1 Flooding

The improved Flooding protocol in our simulation model is implemented in a way different from the traditional flooding approach. In the first place, it is the central controller that serves as the source node delivering flooding packets to all other nodes in a smart home. To do so, the RBDS packets broadcasted by the RBDS originator to smart homes are reframed on the side of the central controller with a new source address (the address of the central controller) and the corresponding packet type (the Flooding type) applicable to the data transmission in a house. In the second place, the implementation of the Flooding protocol is inspired by the approach proposed in [26] to mitigate the network traffic in a smart home. The simplified flowchart of the Flooding protocol is shown as follows:

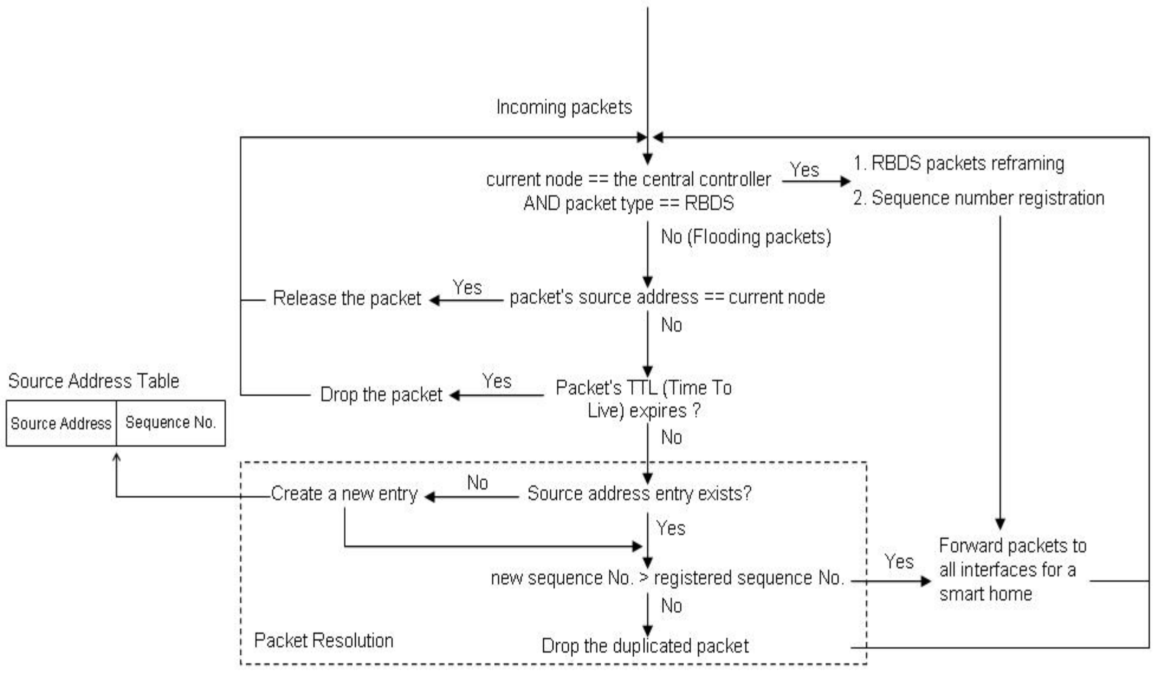

Incoming packets
current node == the central controller AND packet type == RBDS
Yes
1. RBDS packets reframing
2. Sequence number registration
No (Flooding packets)
Release the packet
Yes
packet's source address == current node
No
Source Address Table
Source Address
Sequence No.
Drop the packet
Yes
Packet's TTL (Time To Live) expires ?
No
Create a new entry
No
Source address entry exists?
Yes
new sequence No. > registered sequence No.
Yes
Forward packets to all interfaces for a smart home
No
Packet Resolution
Drop the duplicated packet

Figure 25 A simplified flowchart of Flooding protocol

The normal operation of the Flooding protocol is achieved mainly by a source address table maintained by each node in a house. Unlike generic routing tables, the source address table only consists of two fields: the address of a source node (namely the central controller) that issues flooding packets, and the sequence number of the latest flooding packet. Initially, the central controller transforms incoming RBDS packets into outgoing flooding packets and transmits them to all other nodes in a house after increasing their sequence number by 1. Upon reception of the first flooding packet, a node establishes an entry in its source address table with the source address and the sequence number specified in the header of the packet. Otherwise, the node compares the sequence number in the newly received packet with the registered one in the entry to identify if it received an entirely new flooding packet originated from the central controller. To be specific, a flooding packet with the sequence number greater than the registered one is considered a new one that is used to update the entry and transmitted through all interfaces equipped in the node to other nodes to afterwards, whereas a flooding packet with the sequence number less than or equal to the registered one is directly dropped as a duplicated packet. In other words, each flooding packet issued by the central controller is received only once at all destinations regardless of the interfaces as well as the underlying networks. Those extra packets with the identical sequence number are unanimously intercepted by nodes in a house, which promotes the network performance to some extent.

7.2.2 Timer-Driven Sequential Packet Forwarding

To forward incoming RBDS packets via the ZigBee or AODV routing protocol, routes from the central controller to destination nodes should be established in advance for long-term RBDS packets transmission by disseminating RREQ packets to neighboring nodes so as to calculate the minimal route cost along the way to destination nodes, as illustrated in Figure 26.

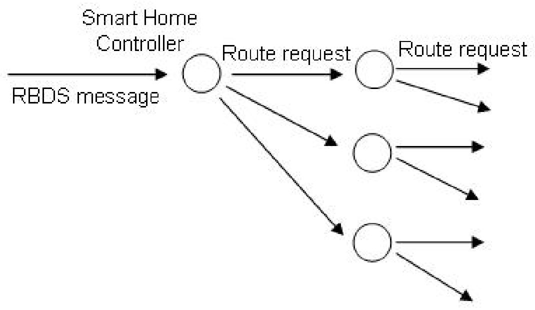


Figure 26 Diagram of route discovery originated from central controller

With the increase of nodes preconfigured to receive RBDS messages, a broadcast-based storm [27] caused by RREQ packets oriented towards distinct nodes could deteriorate the entire network traffic, given that all RREQ packets originated by the central controller are propagated simultaneously to nodes within its signal range, which is not supported in the existing implementation of NS-2. Meanwhile, nodes in a network are most likely to be overwhelmed by consecutive rebroadcasted route quests to different destination nodes and ultimately fail to transmit route reply (RREP) packets destined to the central controller, which in turn severely compromises the data communication between the central controller and the recipients of RBDS messages.

A variety of schemes were proposed by the authors in [28] to address the issue of broadcast-based storm, including the probabilistic scheme, the distance-based scheme, the location-based scheme, the cluster-based scheme, and the counter-based scheme. These schemes are distinguished by their approaches to the estimation of broadcast redundancy as well as the corresponding decision-making. Meanwhile, the authors argued that a stand-alone counter-based scheme is capable of eliminating most of the redundant rebroadcasts in a densely distributed network as compared to the simple flooding approach.

Technically, these schemes accompanied by additional methods are susceptible to special conditions or complicated in terms of software simulations. To be specific, the probability of rebroadcasting statistically varies with a preset random value and the network size in the probabilistic scheme; the distance-based scheme resorts to the signal strength of received packets to approximately estimate the distance of nodes for the decision of rebroadcasting, while the location-based scheme requires the geometric location of nodes in order to minimize rebroadcasting with the help of positioning equipment, such as Global Positioning System (GPS) receivers; the cluster-based scheme forces all nodes engaged in a network to be interconnected so as to form a cluster composed of a head and a group of members by periodically advertizing their presence to neighbors, in which case the role of each node determines whether the node is qualified to rebroadcast packets along the cluster; the counter-based schemes suggested by the authors assumes that all broadcasting packets are tagged with the same broadcast identifier and destined to only one node when comparing the times of received packets with a preset counter threshold, which is infeasible in the case of multiple RREQ packets concurrently issued to various nodes in a network, given the complexity of tracking packets in implementation.

To optimize the processing of RREQ packets and alleviate the issue of broadcast-based storm in our simulation model without fundamentally affecting the nature of route discovery of the ZigBee and AODV protocol, we implemented an approach of timer-driven sequential packet forwarding to the RBDS packet transmission by exploiting the relationship of precedence between RREP packets to originators and data packets to recipients at the routing layer. According to the mechanism of both the ZigBee and AODV routing protocols, an originator has to accept a RREP packet from a recipient for the creation of a route in its routing table before delivering any data packet to the recipient. Considering that all nodes preset to receive RBDS packets are piled up in the transaction list of the central controller, a RREQ packet destined to the next node held in the list is normally postponed till a RBDS packet with an established route to the target node is successfully sent out by the central controller, or is triggered by a timer indicating that the central controller fails to receive the RREP packet from the target node at preset intervals, depending on the numerical order of nodes configured through scenario scripts.

The operational mechanism of sequential packet forwarding is designed on the basis of the packet queuing management existing in the implementation of the ZigBee and AODV protocols, as illustrated in Figure 27.

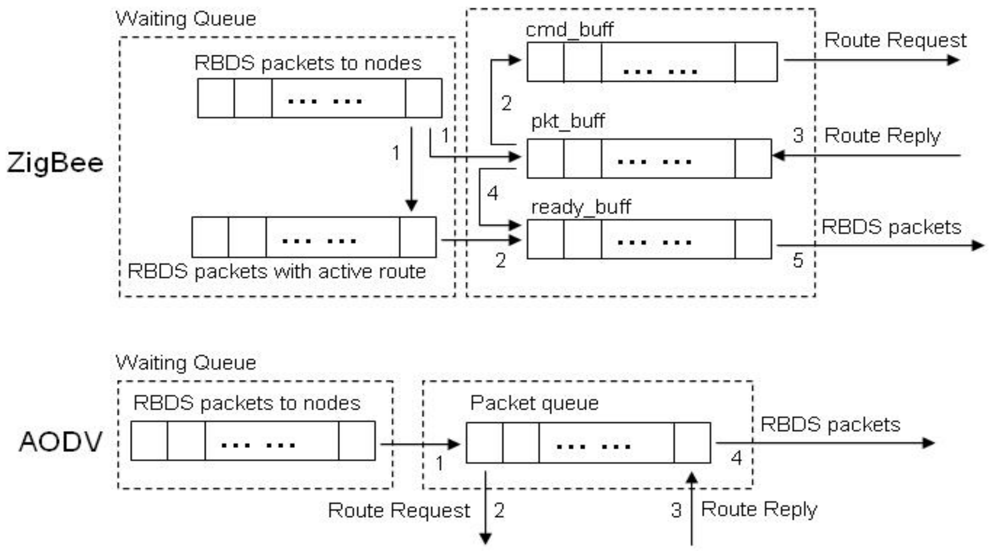


Figure 27 Packet queuing management of the ZigBee and AODV routing layer

Originally, the ZigBee routing object holds three kinds of queues (pkt_buff, ready_buffer and cmd_buff) which are used for data packets without route to destinations, data packets ready to be transmitted and routing command packets respectively, while the AODV routing object holds only one queue intended for data packets. For both of them, routing requests are generated by the routing object for outgoing data packets. The only difference between them is that the ZigBee routing object supplies more than one queue for packets with different status or type in coordinating their transmission triggered by the notification of active status from the MAC layer, as compared to the AODV protocol. Meanwhile, the ZigBee routing object moves data packets destined to a recipient from the pkt_buff to the ready_buff pending for transmission after receiving the RREP packet from the recipient, whereas the AODV routing object directly delivers the data packets extracted from its pending queue. Based on the internal queuing mechanism of packet delivery, queues equivalent to the existing ones are implemented for data packets so as to minimize RREQ packets destined to distinct nodes that occur simultaneously in a network. The corresponding modification to the existing mechanism is shown as follows.

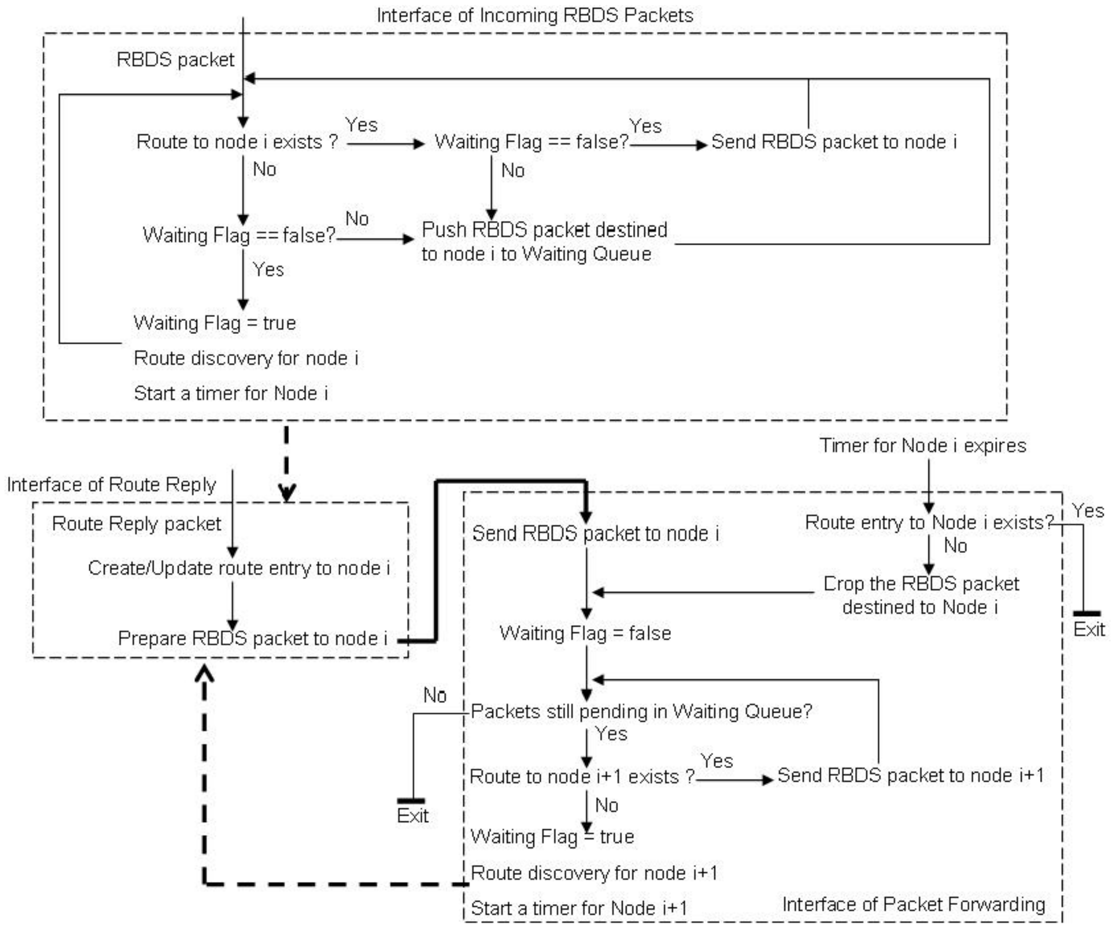

Interface of Incoming RBDS Packets
RBDS packet
Route to node i exists ?
Yes
Waiting Flag == false?
Yes
Send RBDS packet to node i
No
No
Waiting Flag == false?
No
Push RBDS packet destined to node i to Waiting Queue
Yes
Waiting Flag = true
Route discovery for node i
Start a timer for Node i
Interface of Route Reply
Route Reply packet
Create/Update route entry to node i
Prepare RBDS packet to node i
Timer for Node i expires
Send RBDS packet to node i
Route entry to Node i exists?
Yes
No
Crop the RBDS packet destined to Node i
Exit
Waiting Flag = false
No
Packets still pending in Waiting Queue?
Yes
Route to node i+1 exists ?
Yes
Send RBDS packet to node i+1
Exit
No
Waiting Flag = true
Route discovery for node i+1
Start a timer for Node i+1
Interface of Packet Forwarding

Figure 28 Simplified implementation flowcharts of timer-driven sequential packet forwarding

With support of the extra waiting queue extended for the routing object, the sequential packet forwarding is implemented in three interfaces: the interface of incoming RBDS packets from the RBDS network, the interface of RREP packets, and the interface of packet forwarding. At the beginning of the transmission, all reframed RBDS packets to destination nodes are inserted into a waiting queue. Only one RBDS packet is extracted each time by the routing object from the head of the waiting queue till it is successfully sent out to a recipient or is dropped immediately at the expiration of a timer preset for it in that the requested route to the destination remains inactive. As a consequence, the routing object repeatedly executes the same process as the previous one by relocating a RBDS packet for the successive node from the waiting queue to the existing queue.

Technically, the routing object determines whether to send the outgoing RBDS packet by querying the routing table about its active route entry with the corresponding destination address. In such case, the waiting queue is sequentially retrieved in order to get the target RBDS packet through to the existing queue held by the routing object as long as a switching flag is set to be active. In addition, all types of queue at the routing layer are emptied upon expiration of a route counter intended for pending packets in the sense that the remaining RBDS packets queuing up for transmission are considered useless once the routing object receives a new one from the RBDS network from the perspective of the response-demand program.

7.3 Routing Strategies with Multiple Interfaces/Channels

In principle, the existing approaches to MIMC and ours differ with respect to experiential scenarios as well as the objective of implementation. To be specific, the majority of existing approaches place emphasis upon the enhancement of throughput in a heavily loaded wireless network so as to achieve higher performance by fully exploiting the remaining non-overlapped channels available for use in the network. As a consequence, one of the main responsibilities of the routing protocol in these approaches is to strategically choose one from the existing interfaces with support of a certain switching mechanism to minimize the probability of access contention for one channel. One typical example is [29], in which the authors adopted a heuristic routing algorithm in the Dynamic Source Routing (DSR) protocol by computing the cost in interface switching, choice of different channel and the global resource usage at the expense of network delay.

In contrast to them, our solution is implemented as a combination of wireless and wired networks with comparatively low traffic, since short-sized packets are transferred to smart homes from the RBDS network at long intervals due to the characteristics of demand-response programs as well as sporadic activities of nodes associated with control-based operations in smart homes. Furthermore, it is unnecessary for power-restrained devices to increase extra channels in one network from the perspective of the utilization of channels in data transmission.

Given the AODV and ZigBee routing protocols tailored to our requirements, a range of routing strategies are separately introduced so as to measure the performance of the combination of multiple networks from different aspects. With the exception of the wireless link routing and the reliability-based routing that could be directly configured in the scenario scripts, the remaining routing strategies are implemented in particular to address different types of routing alternatives in simulations.

7.3.1 Joint-Path Routing Strategy

To successfully transmit data packets to a destination node in the combined network, the routing object of the source node is capable of establishing a joint route that may traverse multiple networks to the destination node. Meanwhile, the establishment of the joint route is independent of the type of node, in which case any node (the central controller or the remaining nodes in a smart home) could initially launch a joint route over multiple networks or take advantage of the established joint route to the destination that facilitates both the RBDS packet transmission forwarded by the central controller to nodes in a smart home and the status updating issued from any node installed indoors to the central controller.

7.3.1.1 Routing Options across Multiple Networks

In terms of the implementation of a routing protocol, both the ZigBee and the AODV routing object employs a routing table to keep track of the optimal paths destined to recipients in data transmission. For each node in a network, the routing table is composed of a group of active route entries, each of which includes such route-related information as a destination address, a sequence number of route packets, the minimal route cost to the destination node, etc. Given the routing tracking method, we take advantage of route entries in the routing table to establish joint routes that may traverse multiple networks to destination nodes without influencing the existing routing functionalities in the ZigBee and AODV protocol, as illustrated in Figure 29.

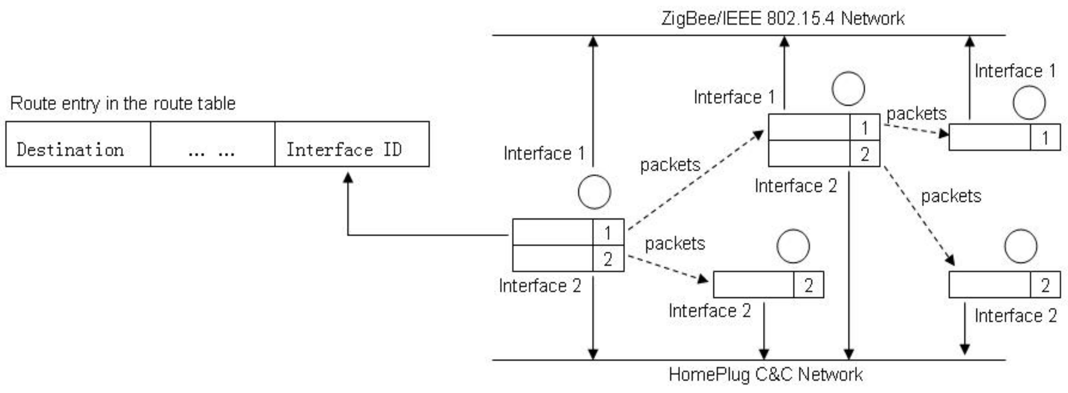


Figure 29 Interface tagging in the route entry for nodes

In our implementation, the combination of wireless and wired networks in a smart home is treated as a single network from the perspective of the routing layer which operates as usual with no knowledge of the realistic deployment of underlying networks. Each link of the combined network could be chosen by nodes with an identical probability to forward data packets. Hence, it is essential that the routing object of each node register the corresponding interface ID in an active route entry for a route with the minimal route cost to a destination node by investigating incoming route packets tagged with the underlying interface ID at the MAC layer. In other words, an extra field is extended in the route entry class so as to record interface IDs required in packet forwarding across multiple networks. To clarify the approach to interface registration, Figure 30 shows the key steps of how to deal with incoming route packets prior to the transmission of data packets.

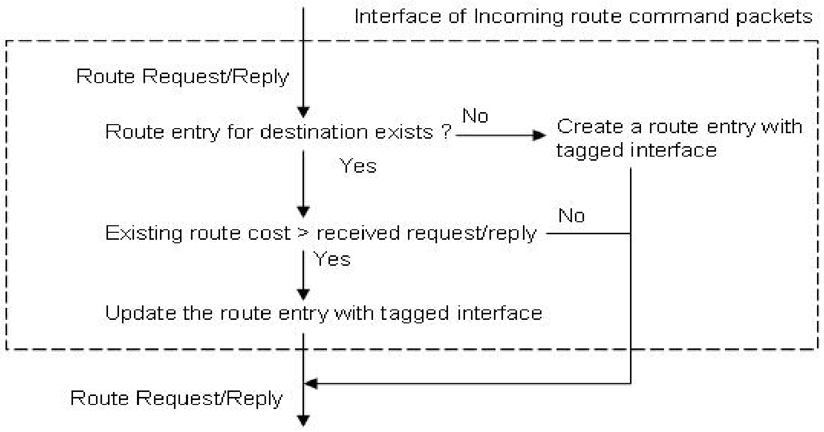

Interface of Incoming route command packets
Route Request/Reply
Route entry for destination exists ?
No
Create a route entry with tagged interface
Yes
Existing route cost > received request/reply
No
Yes
Update the route entry with tagged interface
Route Request/Reply

Figure 30 A simplified flowchart of interface tagging intended for route establishment

RREQ and RREP are two route commands exclusively used for route establishment in the

ZigBee or AODV routing protocols. Upon reception of either of the two types of packet, the routing object of any node in a smart home extracts the interface ID identified in the packet header and decides whether to create a new route entry with the interface ID by examining the availability of a route entry destined to the recipient in the routing table. Besides, an active route entry could be updated with a new path along with the extracted interface ID attached to a different network after comparing the existing route cost in the route entry with the route cost contained in the incoming route packet.

7.3.1.2 Routing with Multi-Interfaces

Generally, data transmission in the implementation of ZigBee or AODV is categorized into two phases: the establishment of routes to destination nodes and data packet forwarding, as illustrated in Figure 31.

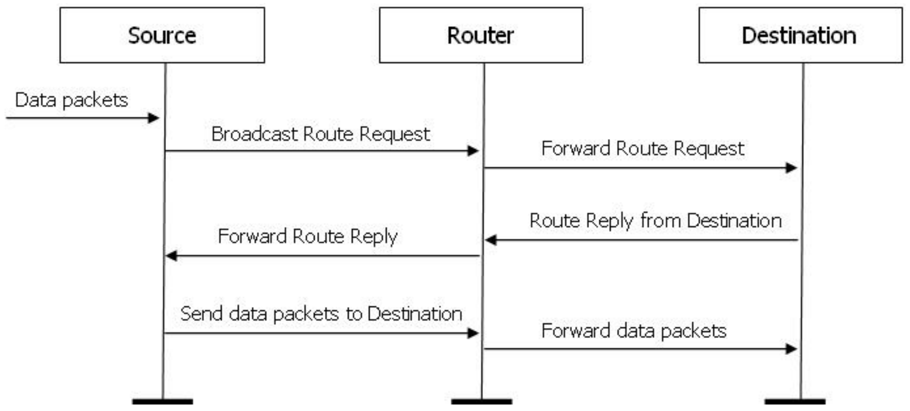


Figure 31 The sequence diagram of data transmission

Upon receiving a data packet handed down from the upper layer, the routing object of a source node in the network issues a RREQ packet to all the neighboring nodes if the route entry to the destination node specified in the data packet header does not exist in the routing table. The remaining intermediate nodes in the network keep refreshing the path from the originator with the minimal route cost in the corresponding route entry in such a way as to establish reverse routes to the source node, when forwarding rebroadcasted RREQ packets from its neighboring nodes to the destination node. As a consequence, a RREP packet unicasted by the destination node travels along reverse routes to the source node as soon as a RREQ packet featured with better route cost arrives at the destination node. Eventually, a single route is established between the originator and the recipient, the pending data packet could be delivered by the routing object along the route to the recipient.

It is evident that packet forwarding within a node is classified into two modes: broadcast and unicast. The broadcast mode refers to the RREQ packet forwarding, whereas the unicast mode includes the RREP packet forwarding and generic data packet forwarding. For the combination of multiple networks in our project, the modes of packet forwarding are closely associated with the underlying interfaces attached to nodes, as shown in Figure 32.

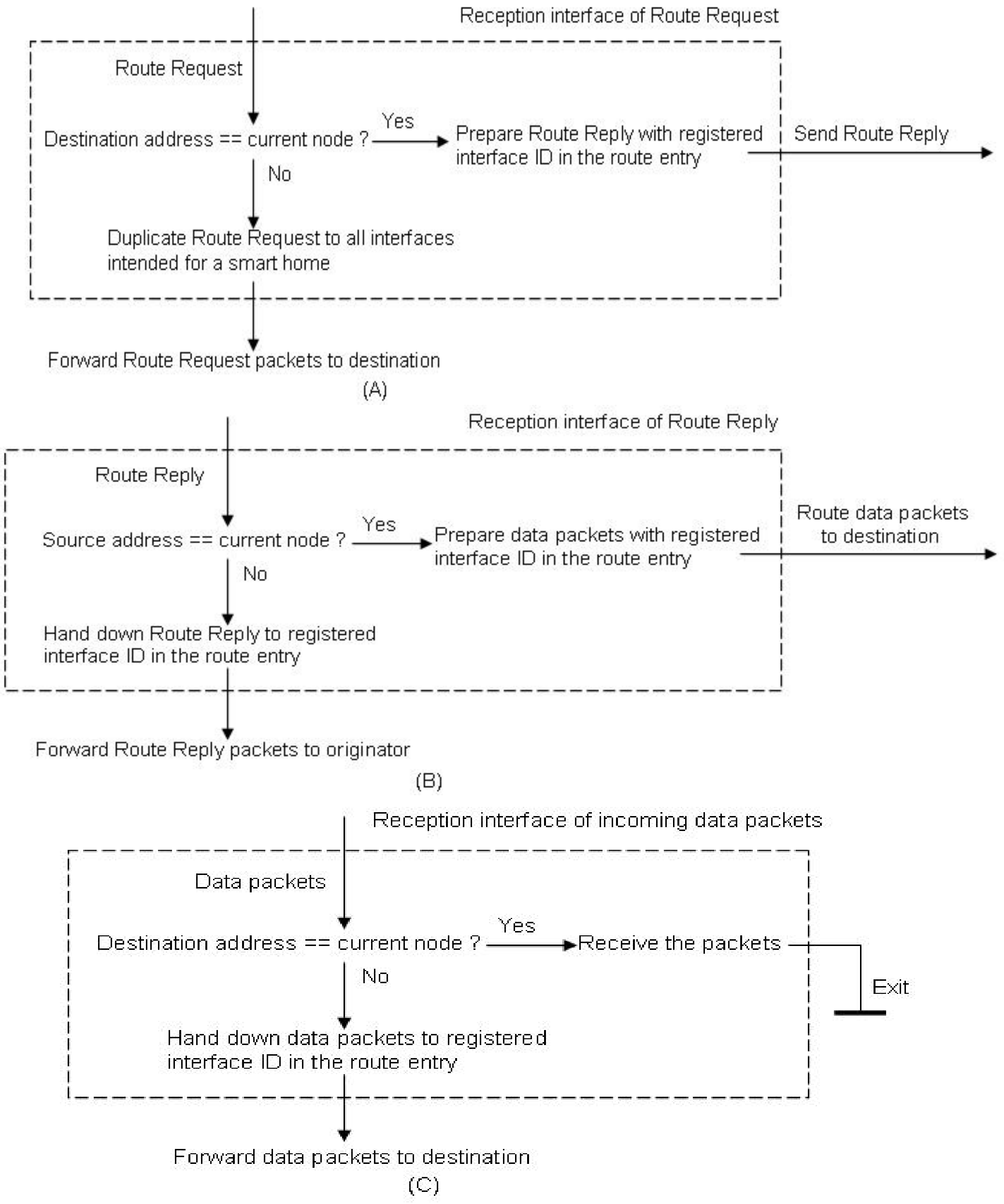

Reception interface of Route Request
Route Request
Destination address == current node ?
Yes
Prepare Route Reply with registered interface ID in the route entry
Send Route Reply
No
Duplicate Route Request to all interfaces intended for a smart home
Forward Route Request packets to destination
(A)
Reception interface of Route Reply
Route Reply
Source address == current node ?
Yes
Prepare data packets with registered interface ID in the route entry
Route data packets to destination
No
Hand down Route Reply to registered interface ID in the route entry
Forward Route Reply packets to originator
(B)
Reception interface of incoming data packets
Data packets
Destination address == current node ?
Yes
Receive the packets
Exit
No
Hand down data packets to registered interface ID in the route entry
Forward data packets to destination
(C)

Figure 32 Simplified flowcharts of handling broadcast and unicast packets

The allocation of interfaces for a node depends on the functionalities of nodes in a smart home. Generally, the central controller equipped with three interfaces across multiple networks transfers RBDS messages from the RBDS network to a smart home and receives the status information from nodes in the house as well, while the intermediate nodes equipped with two interfaces and generic nodes equipped with one interface route packets to recipients or issue status update messages to the central controller.

Technically speaking, a packet is selectively handed down by a routing object within a node to the corresponding interfaces, depending upon the packet type and the total number of interfaces attached to the node. For one thing, broadcast-based RREQ packets are relayed through the ZigBee/IEEE 802.15.4 interface and the HomePlug C&C interface that belong to the central controller and intermediate nodes, and through the ZigBee/IEEE 802.15.4 interface or the HomePlug C&C interface connected to generic nodes, in which case RREQ packets are disseminated via the combination of the two networks in a redundant way; for another, unicast-based RREP packets as well as generic data packets are guided to travel along a single route, with only one interface ID bound with the optimal route cost registered in the routing table of each node. To sum up, the classification of packet forwarding mode separately manipulated in the routing object within nodes enables originators to transfer packets along unique interface-registered routes to recipients, which applies to both the central controller and the remaining nodes in a smart home.

7.3.2 Backbone-Based Routing Strategy

In addition to the data transmission to destinations only through the wireless link, our simulation model offers the HomePlug C&C network as a wired backbone accessible to all end nodes for packet forwarding, since it is convenient for intermediate nodes with both wireless and wired interfaces to be deployed nearly anywhere power outlets are available in a house. For one thing, nodes are enabled to transmit packets over the wired network as much as possible if it is reachable within the range of nodes, which alleviate the potential disturbance produced by nodes through the wireless interface; for another, such routing mechanism could effectively reduce the overall power usage consumed by all indoor nodes during data transmission, given that the central controller and intermediate nodes equipped with multiple interfaces always treats the power line as their own energy source. Hence, we designed such routing strategy based on the HomePlug C&C backbone for the purpose of performance measurement in simulations, as illustrated below.

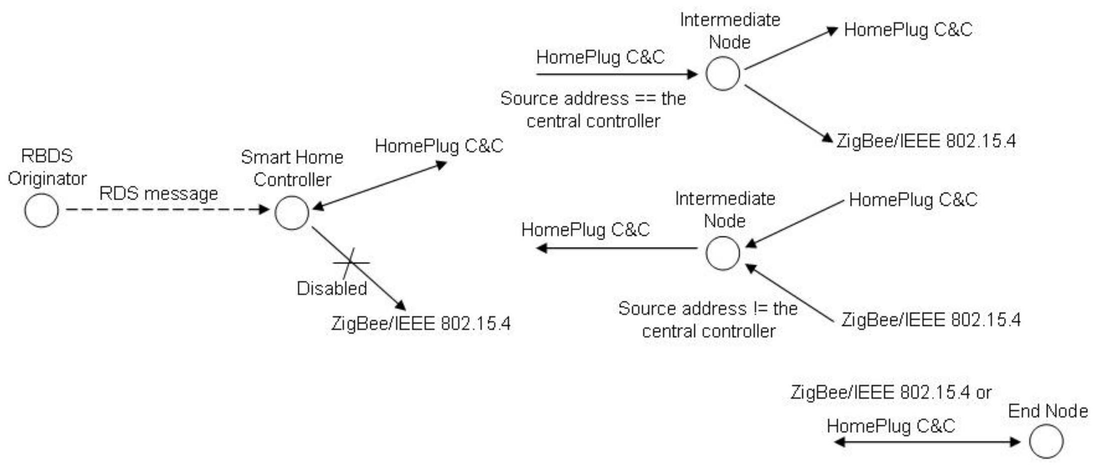


Figure 33 Packet forwarding logic with backbone-based routing

Following the principle of timer-driven sequential packet forwarding intended for RBDS packets, the backbone-based routing strategy is essentially built up on the basis of the joint-path routing mechanism with particular constraints on the wireless interface, in the sense that packets are possibly forwarded by generic nodes equipped with only one wireless interface connected to the ZigBee/IEEE 802.15.4 network. To establish a route to a destination node through the backbone, a RREQ packet is extended with a source interface field in their header that records the count of interfaces hold by the source node when being transmitted. Under such circumstances, intermediate nodes are able to identify the type of source node by examining the source interface field in each RREP packet header (the central controller holds three interfaces; intermediate nodes normally hold both two interfaces; the remaining nodes in a house only hold one wireless interfaces) and determine which interface to transmit the RREP packet afterwards.

Meanwhile, there are two rules implemented in both the AODV and ZigBee routing protocols: firstly, the central controller with the wireless interface disabled in the routing object is only allowed to transmit packets through the HomePlug C&C network; secondly, RREQ packets originated from the central controller are transmitted by local intermediate nodes through both two interfaces, which enable intermediate nodes and end nodes to receive the packets in a wired or wireless way from nodes within their range, whereas RREQ packets issued by intermediate nodes or end nodes will be forwarded by intermediate nodes only through the wired interface to the backbone upon reception. In other words, RBDS packets destined to end nodes are forced to traverse the backbone to the wireless network in which an end node receives RBDS packets directly from a intermediate node reachable to it or indirectly from neighboring end nodes according to their location in the multiple networks; on the contrary, the transmission of status update message destined to the central controller are restricted to the backbone as long as they are intercepted by any intermediate node without regard to the network property of incoming interface. In addition, any incoming RREQ packets issued by nodes with more than one interface will be discarded by local intermediate nodes as long as these packets are received through the wireless interface, which guarantees that routes are always established through the backbone.

7.3.3 Dual-Path Routing Strategy

To achieve higher possibilities of packet reception at destinations with the increase of nodes deployed in multiple networks, it is feasible that the routing object within nodes is specially adjusted so as to allow nodes to take advantage of two entirely distinct routes through both the wireless network and the backbone simultaneously for data transmission. In contrast to the routing strategies characterized with a single path to destination nodes, the dual-path routing strategy provides nodes with another alternative to robust and redundant data transmission, in which case a packet accidentally dropped for some reason along one route has no impact upon its duplicate forwarded along the other route in that the two routes are established independently from each other within each node, except nodes only attached to the HomePlug C&C network. Eventually, a destination node receives two identical data packets issued from a source node. Hence, we designed and implemented the mechanism of dual-path routing in both AODV and ZigBee routing protocols as illustrated in Figure 34.

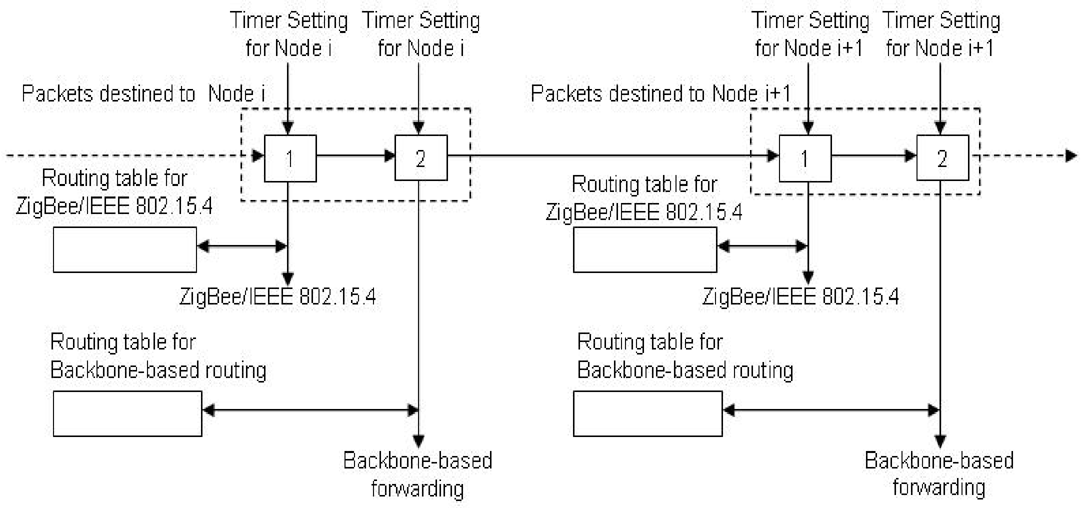

Timer Setting for Node i
Timer Setting for Node i
Timer Setting for Node i+1
Timer Setting for Node i+1
Packets destined to Node i
Packets destined to Node i+1
1
2
1
2
Routing table for ZigBee/IEEE 802.15.4
Routing table for ZigBee/IEEE 802.15.4
ZigBee/IEEE 802.15.4
ZigBee/IEEE 802.15.4
Routing table for Backbone-based routing
Routing table for Backbone-based routing
Backbone-based forwarding
Backbone-based forwarding

Figure 34 Simplified diagram packet forwarding with dual routing

Technically speaking, the dual-path routing strategy is considered an integration of the wireless link routing strategy and the backbone-based routing strategy, in the sense that the transmission of each data packet and its duplicate originated from source nodes are determined by the two routing strategies respectively. To achieve so, we created two routing tables within the routing object for each node preconfigured with the dual-path routing strategy in a house, each routing table corresponding to one of the two base routing strategies. With the support of the wireless link routing strategy, nodes equipped with multiple interfaces in a smart home transmit packets through the wireless interface to ensure that packets are disseminated only in the ZigBee/IEEE 802.15.4 network. Also, the backbone-based routing strategy preset in the routing object are performed individually to force nodes to forward packet through the wired interface as much as possible. In this way, two routes to each destination node are generated independently in the two routing tables within all nodes involved in the networks.

In addition to the integration of the two routing strategies, the existing mechanisms are slightly adapted to the situation of dual-path packet forwarding. First of all, each of two RREQ packets featured with different routing strategies intended for the same data packet are configured with one timer when being sent. Under such circumstances, the timer tied to the first route will trigger the other RREQ packet transmission in an attempt to establish the second route upon expiration, which indicates failure to receive the RREP packet relevant to the first route. Given the mechanism of packet queuing management, the routing object within the central controller will repeat the same procedure for the next destination by moving another RBDS packet to the pending queue after completing the transmission of the previous data packet or automatically triggered by the timer of the previous one tied to its second route. By contrast, the routing object within generic nodes could remain idle till it receives the next status update message passed down from the application layer at regular intervals if configured in the scenario scripts. Secondly, a new field labeling the type of routing strategy is inserted in the header of each packet by source nodes before transmission, which contributes to the identification of the routing strategy made by forwarding nodes along the path to destinations. With the mechanism of field tagging, the wireless link routing strategy and the backbone-based routing strategy can coexist and be performed separately in the multiple networks.

7.4 Issues Associated with ZigBee Routing

7.4.1 Adjustments of the ZigBee/IEEE 802.15.4 MAC/PHY Layer

The author of the ZigBee routing implementation updated part of the ZigBee/IEEE 802.15.4 MAC/PHY Layer implementation existing in NS-2 in an attempt to meet the demands of ZigBee routing for the purpose of their own analysis. To fit in with our simulation model, a group of adjustments are made to ZigBee/IEEE 802.15.4 MAC/PHY Layer as follows:

1. Clear Channel Assessment (CCA) is performed by the PHY Layer using energy detection and carrier sense to detect whether the medium is busy or not. The author of the ZigBee routing implementation adjusted the length of CCA to stop any channel activity during the period of CCA in NS-2 v2.30. However, the improvement of CCA in the public release of NS-2 since v2.31 has made such adjustment meaningless in the later versions, in which the PHY layer performed the CCA at the middle of the original period and added a new timer to report the status of the channel at the end of the original period [12]. To conform to the update, we follow the public release of NS-2 by removing the

adjustment in the corresponding source code and scenario scripts.

2. The mechanism of data retransmission, which was originally disabled by the author of the ZigBee routing implementation, is re-enabled in our project through scenario scripts to address the issue of failure to access the communication channel after several CCA periods.

3. The CSMA/CA related parameters redefined by the author for flexible configurations in scenario scripts are disabled in our project so as to keep a fair comparison with the AODV routing protocol, since such parameters are assigned a default value in the existing NS-2 implementation.

4. The author of the ZigBee routing implementation adopted a collision flag already defined in the ZigBee/IEEE 802.15.4 MAC Layer implementation to force discarding all received packets in case they collide with other incoming packets in terms of the reception power. Nevertheless, acknowledgment frames to packet originators could also be indiscriminately abandoned with a higher possibility, which inevitably compromises the overall network traffic in simulations. Hence, the flag is disabled in our project so as to get acknowledgment frames through to the upper layer for further handling.

In addition, incoming packets passed up from the ZigBee/IEEE 802.15.4 MAC Layer are re-directed towards the upper layer through the ZigBee routing layer for the purpose of trace logging as long as the ZigBee routing layer object confirms that these packets arrived at their destination by examining their addressees along with the packet type.

7.4.2 Synchronization Mode of Packet Forwarding

With the mechanism of packet scheduling, an object representing a routing protocol in NS-2 asynchronously manipulates outgoing packets. In other words, the routing object leaves the tasks of packet transmission through layers to the scheduler object by inserting packets into the scheduling queue managed by the scheduler object. It is the scheduler object that takes responsibility of dispatching these packets from the upper layer to the destination objects at the lower layer. As a consequence, the routing object is capable of keeping forwarding packets without the knowledge of whether they are successfully transmitted. Hence, any protocol object within a mobile node could maintain independence from each other without direct intervention of packet delivery in simulations.

Contrary to the canonical approach to packet scheduling at the routing layer, the ZigBee routing protocol was implemented in a synchronized way, in which case the ZigBee routing object takes direct control of the ZigBee/IEEE 802.15.4 MAC layer, bypassing the LL object, as illustrated on the left side in Figure 35.

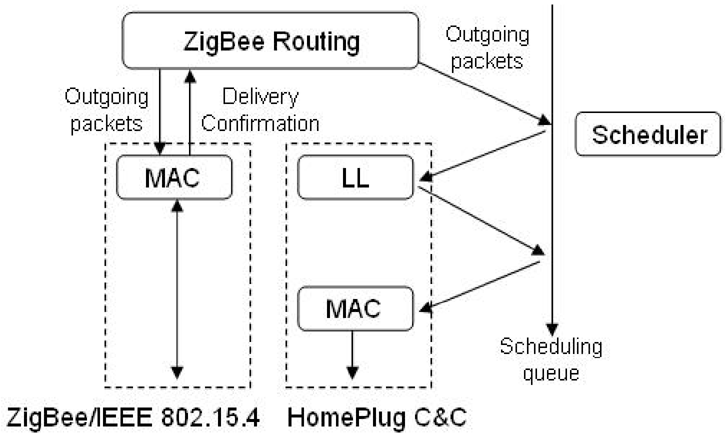


Figure 35 Synchronization of Packet forwarding in ZigBee routing implementation

Specifically, a cross-layer flag set in the ZigBee routing object aims to detect the status of activity at the ZigBee/IEEE 802.15.4 MAC layer. When a packet is passed from the ZigBee routing object to the MAC layer, the flag is set to be busy by the ZigBee routing object. The flag is always kept unchanged till a notification is issued from the MAC layer to the ZigBee routing object, indicating a successful transmission or a failure of access to the shared channel after retransmissions. Subsequently, the ZigBee routing object can pass down the next packet extracted from the data buffer or the command buffer.

The synchronization mode in controlling the ZigBee/IEEE 802.15.4 MAC layer cause problems to our project mainly in three aspects as follows, especially in dealing with packet transmission through multiple interfaces.

1. With the increase of nodes engaged in a network, this approach leads to a deadlock on the receiver side with a higher possibility in that the underlying layer is unable to release the corresponding resources intended for packet transmission. For instance, a destination node fails to transmit a RREP to an originator while keeping listening to RREQ packets, which in turn blocks subsequent communications between them, since the remaining RREQ packets forwarded through other nodes to the destination node could be discarded due to a higher cost.

2. A tradeoff has to be implemented in nodes with both the ZigBee/IEEE 802.15.4 interface and the HomePlug C&C interface, including the central controller and the intermediate nodes, considering that the ZigBee routing object must conform to the asynchronous mode in handing outgoing packets down to the HomePlug C&C MAC layer.

3. Incoming packets from the HomePlug C&C interface are discarded due to the mismatch of packet type in the corresponding LL object, since the ZigBee routing implementation was originally adopted to pass outgoing packets directly to the ZigBee/IEEE 802.15.4 MAC object rather than to the LL object which is bypassed in the ZigBee/IEEE 802.15.4 interface when encapsulating outgoing packets.

4. The PAN identifier in a ZigBee/IEEE 802.15.4 network is used to distinguish the network from other networks in data communication. In the original implementation of the ZigBee routing protocol, the PAN identifier in outgoing packets is forced by the ZigBee routing object with a default value only adopted for a single ZigBee/IEEE 802.15.4 network with nodes numbered from 0, which in turn results in rejection of incoming packets on the receiver side due to the mismatch of the default value with the existing PAN identifier preset in nodes.

In view of the synchronous manipulation inherent in the ZigBee routing implementation, several technical improvements are made to address these issues without fundamentally affecting the operational mechanism of the ZigBee routing layer.

1. The ZigBee routing object forces the underlying layer to release channel resources and resets the cross-layer flag when it detects that the counts of failure to transmission exceeds a preset threshold value hardcoded in our implementation.

2. The implementation of coordinating packet transmission in the case of multiple interfaces is illustrated in Figure 36.

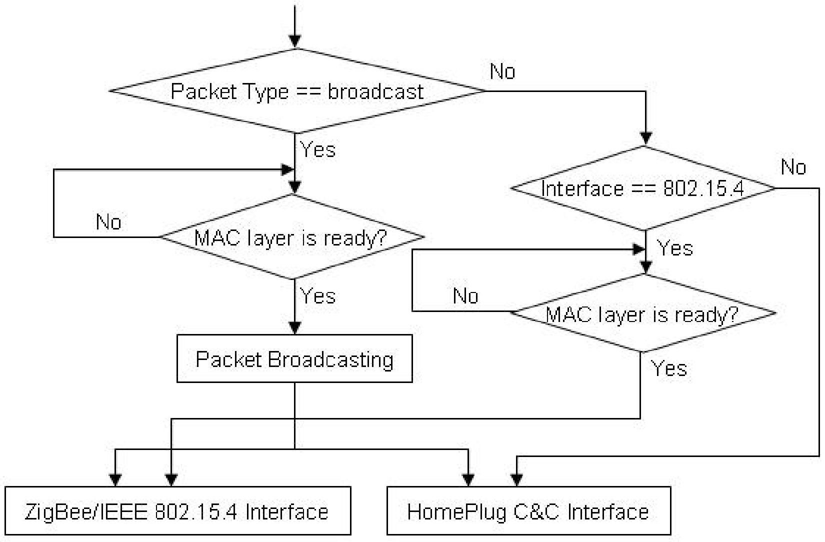


Figure 36 Packet forwarding in nodes with multiple interfaces

For nodes with the ZigBee/IEEE 802.15.4 interface and the HomePlug C&C interface, the direction of outgoing packets is determined by their destination address and the corresponding type at the ZigBee routing layer. On one hand, broadcast packets could be consecutively handed down to these two interfaces; on the other, unicast packets are transferred to a single interface registered in the routing table on the basis of the ZigBee routing strategy. In both cases, the cross-layer flag intended for the ZigBee/IEEE 802.15.4 MAC layer is still kept active so as to enable the ZigBee routing object to interact with both interfaces in parallel in terms of packet forwarding.

3. Packet encapsulation oriented towards outgoing packets passed down from the ZigBee routing object is implemented in the LL object with the HomePlug C&C interface so as to

address the issue of packet type for all packets before transmitting them to destinations.

4. Outgoing packets are reset with the PAN identifier currently available in the ZigBee/IEEE 802.15.4 network before transmission since it is dynamically generated by the PAN coordinator (the central controller) with its node identifier when establishing the network.

8 Application/Transport Layer

The application layer and the transport layer in NS-2 cooperatively contribute to form an application specific to user requirements. An application-layer object is re-defined by users or identified by generic functions that act as a data source and model the user demand in terms of data transmission (i.e. data rate, packet size, interval for transmission, etc), while a transport-layer object (any network object in charge of creating and releasing packets is an agent that acts as a bridge which connects an application and a low-level network in the scope of NS-2 [13]) constructs packets with specific data from the application layer as an input and forwards them to the routing object of the routing layer via an address classifier (or multiplexer). With the help of the routing object, packets are transferred to the destination object of the transport layer and reception applications for further handling.

For short size packet transmissions with no data delivery guaranteed, NS-2 provides the combination of Constant Bit Rate (CBR) traffic for an application-layer object and a User Datagram Protocol (UDP) agent taking responsibility for connectionless data communications. An example of this type is illustrated in Figure 37:

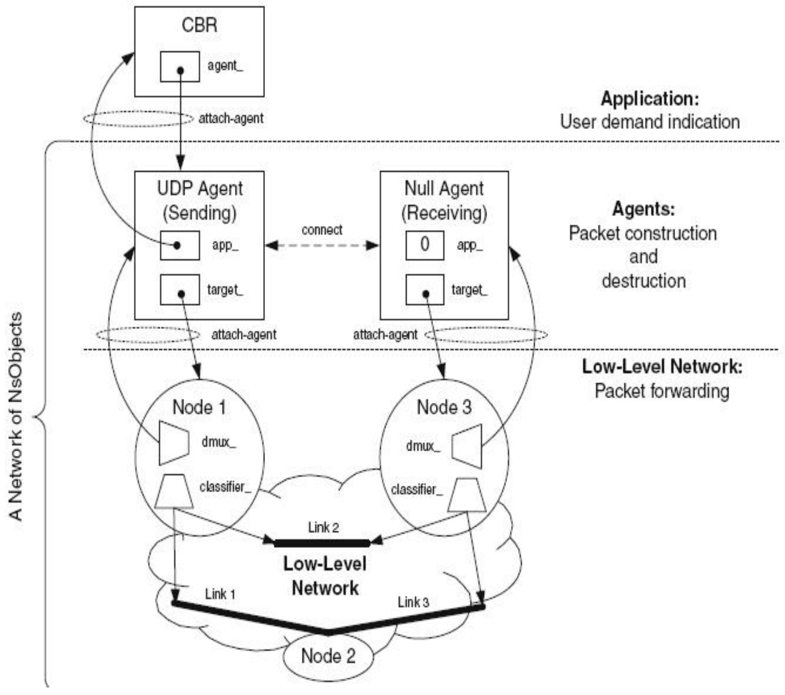


Figure 37 The scenario of CBR over UDP [13]

Node 1 with a CBR application and an UDP sending agent attached acts as a data sender, while node 3 with a null agent attached acts as a data receiver. On the sender side, a CBR application models the user demand to be passed to an UDP sending agent for periodical communication. Subsequently, it is the UDP agent that constructs the packet with source and destination addresses as well as the transport layer port inserted in the packet header, and passes it to the routing agent attached to the node 1. The routing agent mainly delivers packets to destination nodes through the route entry oriented towards the destination in the routing table. Upon reception of packets on the receiver side, the address multiplexer along with the port multiplexer is used to pass packets from the underlying network to a null agent attached to the port identified in the packet header. As a consequence, the null agent directly releases the received packets by reclaiming the memory allocated to the packets since no further processing is required for UDP traffic.

In the scenario of smart home networking simulations, a CBR application over UDP could be employed to simulate the nodes in a house periodically transmitting data to the central controller at the time that RBDS messages are being sporadically forwarded by the central controller to destination nodes pre-configured.

The transmission of RBDS messages is illustrated below:

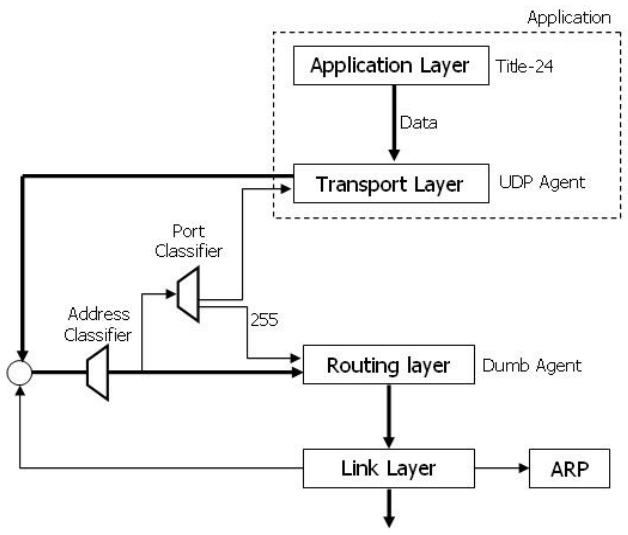


Figure 38 Data packing in RBDS message sender

To simulate messages issued by utilities to smart homes over the RBDS network, a Title-24 specification [10] based application class that extends the Application class in NS-2 is employed to generate Programmable Communicating Thermostat (PCT) application data at fixed intervals specified in scenario scripts with a timer mechanism. In addition to encapsulating the application data from the Title-24 object with addresses and the port number, the UDP agent attached to the RBDS message sender also initializes the packet type as well as the corresponding flag. In this way, RBDS packets could be distinguished from other packets and properly handled upon reception in simulations.

9 Node Addressing Scheme in NS-2

9.1 Issues of Data Transmission with Multiple Interfaces

NS-2 is considered an OO simulation environment in the sense that any entity including a node, layers and channels is treated as an object independent from each other. To identify them in terms of routing or packet forwarding in simulations, each entity involved must be assigned a unique ID as its address in steps of node configuration before the simulation is executed, as illustrated in Figure 39:

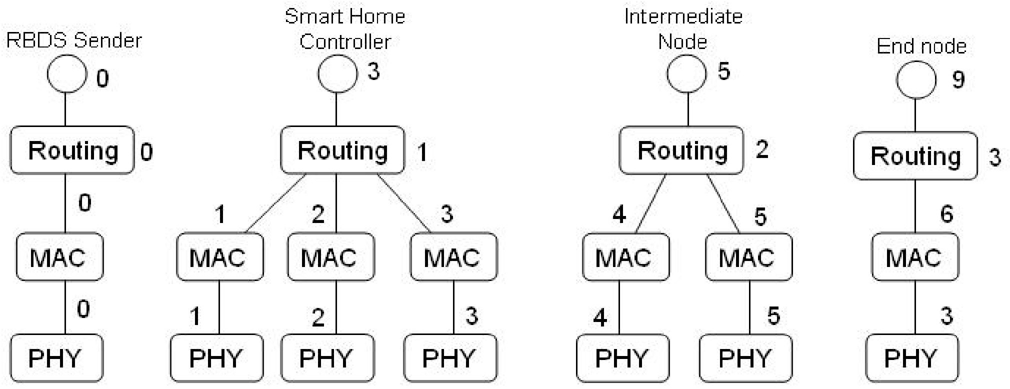


Figure 39 Object ID assignment for each layer in NS-2 environment

For each node, its ID and corresponding address could be assigned automatically from a static integer variable defined in its OTcl class if there is no direct assignment to them or manually if a number is attached to them in simulation scripts (Node ID and its address are identical in NS-2). Following that, the ID/address is passed through the OTcl object of a node to the corresponding C++ object of the node based on the variable binding mechanism in the split model in NS-2.

The ID of Layers covering routing layer, MAC layer and PHY layer are generated automatically from a static integer variable only shared in their peer objects. The static variable for each layer increases by one each time when a value is assigned as a new ID to a layer object. In this case, the ID assignment of nodes and layers originate from different places only for their own use. In other words, there is no direct or indirect connection between node ID/address and layers' ID.

However, the identification of a source address and a destination address intended for data transmission is only set through script files. It is obvious that a random choice of node IDs will cause trouble in traces or log analysis in terms of addresses of nodes, since each layer object only recognizes their own peers when in execution. More than that, it ultimately results in address mismatch and packet loss under the circumstance of multiple interfaces/channels. To be specific, an example of data transmission in the ZigBee/IEEE 802.15.4 network is illustrated in Figure 40:

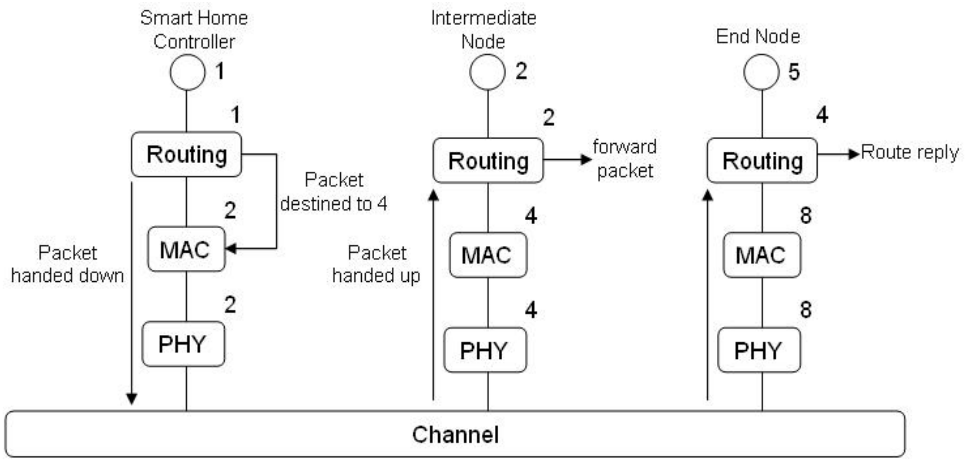


Figure 40 Issues of destination recognition in ZigBee routing

According to the figure, the ID of end node numbered 5 is chosen randomly without consideration of automatic increment of the static integer variable for ID assignment for nodes and layers. When controller node numbered 1 tries to forward a RBDS message destined to node 4 (only known in the script file but unclear about the corresponding ID at the routing layer), it directly sends a broadcast request down to the MAC layer with the routing layer ID as its source. As a matter of fact, only node 5 with routing layer numbered 4 could successfully recognize the destination in the packet transmitted in the network and accept it while the rest of nodes only take responsibility of relaying the request along the way. Even so, it is technically impossible that node 5 sends the RREP packet to the controller numbered 1 as the MAC/PHY layer numbered 1 held by the controller belongs to the RBDS network rather than the ZigBee/IEEE 802.15.4 network.

Ultimately, all packets destined to node 4 are dropped by the central controller at expiration due to a failure in delivering the packet, which in turn deteriorates data delivery of any kind in the whole network.

9.2 United Addressing Solution to Multiple Interfaces/Channels

To avoid such problems caused by mismatch and misuse of the ID/address in the environment of multiple interfaces/channels, a united ID/address assignment scheme is introduced as illustrated in Figure 41:

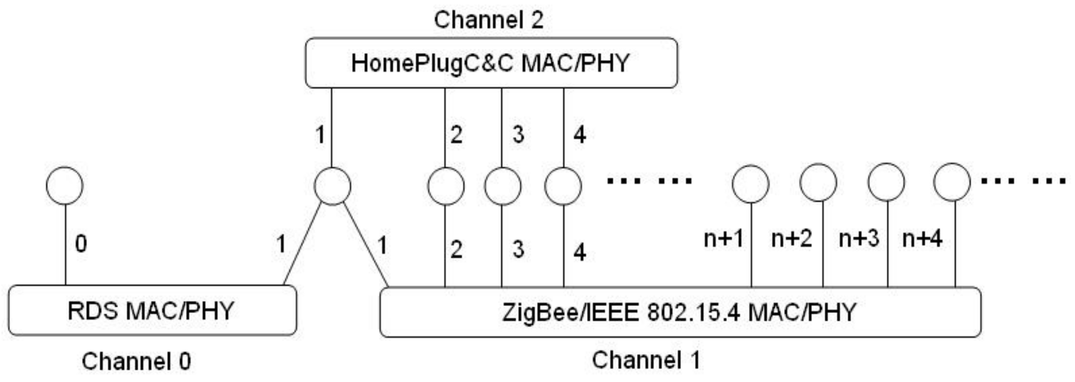


Figure 41 United addressing scheme in the simulation environment with multiple interfaces

In this scheme, any node in multiple networks is manually assigned a unique ID around the simulation environment from the node object, routing layer to MAC/PHY layer regardless of which network it belongs to. The idea behind it is based on the fact that the node ID/address, routing layer ID as well as MAC/PHY layer ID is considered unique within each one of these networks. In doing so, collisions associated with node addressing among multiple networks are eliminated through the multiplexing of ID/address.

In order to keep consist with existing implementations in NS-2 without any impact in terms of address transferring, three steps have been involved to achieve the scheme as Figure 42 shows:

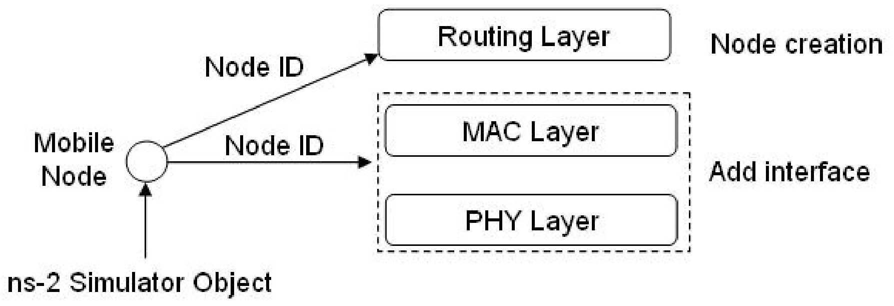


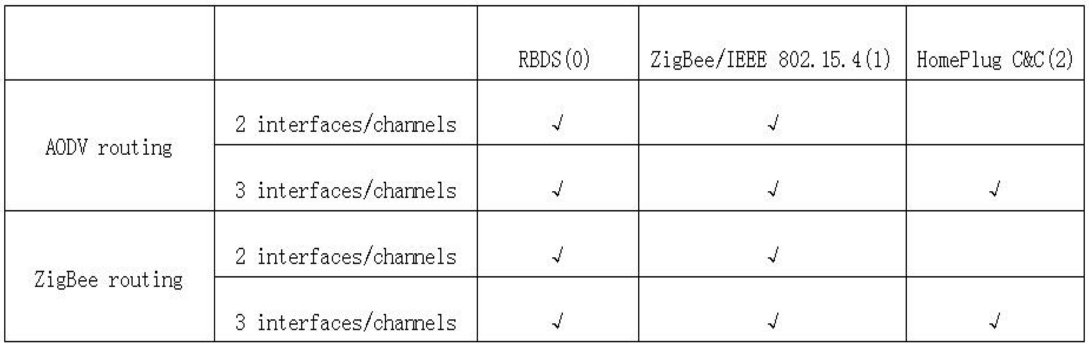
Figure 42 Node ID/Address passing from node to each layer

First of all, node ID is manually assigned to each node involved in the replacement of static assignment before execution. After the node is created, the node ID is passed to its routing layer object through an OTcl method existing in NS-2 that connects the routing layer object to the node object. Following that, the node ID is passed down to the MAC/PHY layer objects after the MAC/PHY layer objects are generated by adding multiple interfaces to the node object. In such way, those IDs that statically increase in each layer object could be overridden with new IDs passed through from the corresponding node.

10 Configuration and Execution of Simulation Script

10.1 Basic Setting and Interface Allocation

The Tcl script file designed for our project allows a combination of multiple network simulations as shown in Table 1:



		RBDS(0)	ZigBee/IEEE 802.15.4(1)	HomePlug C&C(2)
AODV routing	2 interfaces/channels	√	√	
	3 interfaces/channels	√	√	√
ZigBee routing	2 interfaces/channels	√	√	
	3 interfaces/channels	√	√	√

Table 1 Interface setting of AODV/ZigBee routing for multiple networks

For scenarios with both two and three interfaces/channels, the settings of both AODV and ZigBee routing are identical in terms of the allocation of interfaces/channels. In this case, node 0 is identified as the RBDS sender that issues the original RBDS messages to smart homes, whereas node 1 is identified as the central controller in smart homes that is in charge of receiving incoming RBDS messages and forwarding them to destination nodes pre-configured in the network. For the moment, it is assumed that only one smart home is allowed to be configured for receiving RBDS messages from the RBDS network due to the limitation of the software implementation mechanism in our simulation model as well as the deployment of nodes in scenario scripts.

The numbering scheme of interfaces is listed below in Table 2:

Interface	Node Type	Interface Numbering		
		RBDS	ZigBee/IEEE 802.15.4	HomePlug C&C
2 interfaces/channels	RBDS Sender(node 0)	0		
	Central Controller(node 1)	0	1	
	End Nodes		0	1
3 interfaces/channels	RBDS Sender(node 0)	0		
	Central Controller(node 1)	0	1	2
	Intermediate Nodes		0	1
	End Nodes		0	1

Table 2 Interface numbering for different scenarios in simulations

For any node in multiple networks, the interface ID is always gradually allocated starting from 0 regardless of the simulation scenarios. To be specific, nodes with one interface to the wireless network are allocated an interface ID numbered 0, while nodes with one interface active in the backbone are allocated an interface ID numbered 1; nodes with two interfaces are allocated two interface IDs from 0 to 1; nodes with three interfaces are allocated three interface IDs from 0 to 2. Meanwhile, the numbering scheme is forced upon each node involved in networks and directly influences the configuration of interfaces and the creation of nodes, since part of the implementation associated with the interface ID has been hardcoded for our simulations. In such case, misuse of interface ID will lead to unexpected errors during simulations.

10.2 Scenario and Location of Nodes

To generate a scenario of delivering RBDS messages to from the utility to a smart home, location setting is the first element to be considered. Generally, the RBDS sender is located at (0, 0) since it is the first node generated in multiple networks. The active scope of the remaining nodes is restricted to a square of 15 m x 20m as an example to mimic the flat space typical to a house, as illustrated in Figure 43:

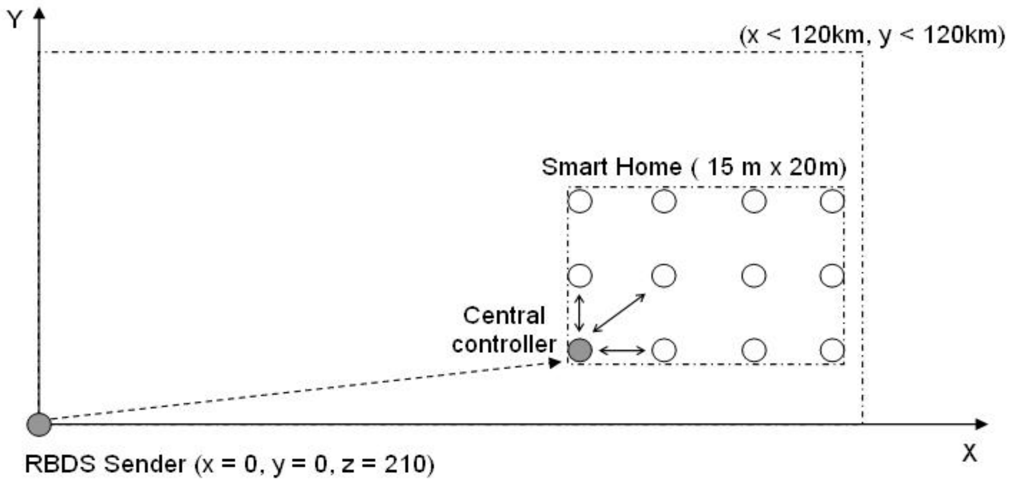


Figure 43 Scenario of RBDS message delivery in smart homes

As for the distance between the coordinate of the RBDS sender and the central controller located in a smart home, the author of the RBDS implementation demonstrated in simulation that devices up to 140 km from the transmitter are capable of receiving RBDS message 30 bytes long in size with a high reception probability of 99.5% [10]. To increase the probability of reception, RBDS messages could be transmitted multiple times (i.e. 5 times for a distance of 120km away from the transmitter).

Based on this result, we choose a distance of less than 120km to minimize the influence of RBDS message broadcasting in terms of packet loss. Meanwhile, the location of all nodes in a smart home has to be readjusted to cope with the situation. For instance, the location of nodes except the RBDS sender could be generated based on the origin of the coordinate system. Then, the X value for each node increases by 100,000 to represent the location readjustment. In this way, the smart home is always kept within the reach of RBDS message delivery.

In addition, Omni-Antenna is chosen for its ability to transmit with equal power in all directions for most wireless communications during simulations as long as the density of nodes is not intentionally restricted to a fixed location or direction. The antenna height of the RBDS message sender is set to 210m for data transmission whereas the antenna height of the central controller in a smart home is set to 1.5m in simulations [10].

Whether the antenna height of the remaining nodes in a smart home remains equal to that of the central controller is unnecessary in that there is no direct influence upon data communication within a smart home. To simplify simulations, the height of all nodes in the house could be set to 1.5m so as to keep consistent with the central controller.

Script sample:

```
#Coordinate of RBDS message sender
$node_(0) set X_ 0
$node_(0) set Y_ 0
$node_(0) set Z_ 219

$node_(1) set X_ 110030.0
$node_(1) set Y_ 40.0
$node_(1) set Z_ 1.5

$node_(2) set X_ 110022.0
$node_(2) set Y_ 32.0
$node_(2) set Z_ 1.5
```

10.3 General Steps in Simulations

Considering the fact that scenarios in simulations deal with in multiple networks, it is necessary to arrange the order of parameter configuration and node creation to avoid problems due to any mismatch or misuse of interface ID. Therefore, the general steps for all scenarios are listed in an ordered way below:

0. Configuration of simulation environment

1. Parameter configuration of each layer intended for node 0 (RBDS transmitter)

2. Creation of node 0 with one interface

3. Parameter configuration of each layer intended for node 1 (Central Controller)

4. Creation of node 1 with multiple interfaces

5. Configuration of nodes for reception of RBDS message

6. Parameter configuration of each layer intended for intermediate nodes (if three networks involved)

7. Creation of intermediate nodes with two interfaces

8. Parameter configuration of each layer intended for normal nodes

9. Creation of normal nodes with one interface

10. Agent setting for all nodes involved

11. Establishment of communication from the RBDS transmitter to the central controller.

12. Establishment of communication from the nodes in a smart home to the central controllers (optional but not necessary)

13. Execution of simulation

It is to be noted that all steps involved have to be strictly followed in the order listed above, especially for the steps of node configuration. Any change to the order among steps could lead to unexpected errors during execution.

Those steps in italic type represent steps where special attentions should be paid in writing simulation scenarios in script files in that they are closely interconnected with the implementation specific to node configuration and communication among nodes. Therefore, all the steps involved including general settings specific to a certain layer are explained here in detail (step 10 and 11 are discussed later in Section 10.5) in order to facilitate the configuration of scenarios.

1>Configuration of simulation environment

At the beginning of the scenario script file, the number of interfaces and the number of wireless channels involved have to be initialized for current scenarios so as to be used as arguments in the configuration of node and corresponding interfaces.

Script sample:

```
set val(ni)          3           ;#number of interfaces
set val(nc)          2           ;#number of wireless channels

set val(chan)        Channel/WirelessChannel    ;#Channel Type

# Create wireless channel
```

```
#set chan_ [new $val(chan)]
for {set i 0} {$i < $val(nc)} {incr i} {
        set chan_($i) [new $val(chan)]        ;#create a channel object for each interface
}
```

In the segment of script listed above, val(nc) is always set for RBDS and ZigBee/IEEE 802.15.4 network. For each network in the scenario, a communication channel object independent from each other has to be generated with a "for" loop to group the nodes attached to it. As for the power line channel intended for HomePlug C&C network, a brand-new channel object is created since it is cloned from Channel/WirelessChannel (detailed in HomePlug C&C in Section 6.4).

Script sample:

```
# Power line Channel Type
set val(pwrchan)        Channel/PowerChannel

# Create power line channel
set pwr_chan_ [new $val(pwrchan)]
```

In NS-2 simulations, a General Operations Director (GOD) instance is generated to store connectivity information (i.e. hop counts) of the topology by creating a matrix with the total number of wireless nodes passed as arguments. Thus, the initialization of GOD has to be updated in our project to ensure the memory management appropriate to nodes with multiple interfaces.

Script sample:

```
set val(mn)          14              ;# number of mobilenodes in the whole network

# Create God
set god_    [create-god [expr $val(mn)*$val(ni)]]
```

2>Creation of node 0 with one interface

Given the restriction in our implementation, node 0 is chosen to be the RBDS sender and node 1 is hardcoded to be the central controller installed in a smart home for the purpose of simplicity in the indexing of the remaining nodes.

After generation, the NS-2 simulator object enables nodes to initialize their parameters during node configuration. In this way, the number of interfaces has to be passed through for later modification when adding interface to nodes.

Script sample:

```
#configure and create node_(0) with DumbAgent routing in RDS network
$ns_ node-config -adhocRouting $rp_(0) \         ;#routing protocol /Dumb Agent
            -propType $prop_(0) \                ;#propagation model
            -topoInstance $topo \                ;#topology
            -agentTrace ON \                     ;#trace log at application layer
            -routerTrace OFF
            -macTrace OFF \
            -movementTrace OFF \
            -channel $chan_(0) \
          -ifNum    $val(ni)     ;#initialize the number of interfaces before the creation of node
```

Following that, the ns simulator resets the simulation environment for node 0 by changing the interface number to 1 since node 0 is only connected to the RBDS network. Meanwhile, the routing protocol is set to Dumb Agent intended for one-hop routing from node 0 to the smart home. Then, parameters of the node interface including the type of LL, the type of MAC layer, the queue type of the LL interface queue, the length of the queue, the type of antenna, the type of propagation model as well as a channel instance is passed through before the creation of node 0. Among them, the propagation model will be generated as an instance in order to be attached to the corresponding PHY layer in that each PHY layer of nodes with multiple interfaces holds a propagation model connected to it for private use in simulations.

Script sample:

```
#Add one interface for node_(0) in RDS network
$ns_ change-numifs    1
```

```
$ns_ add-ifs 0 $ll_(0) $mac_(0) $ifq_(0) $ifqlen_(0) $ant_(0) $prop_(0) $netif_(0) $chan_(0)

set node_(0) [$ns_ node 0]    ;#specify the ID of node as 0
$node_(0) random-motion 0    ;#disable random motion
```

3>Creation of node 1 with multiple interfaces

Similar to node 0, the routing protocol is set to Flooding, AODV or ZigBee routing for node 1 (the central controller) within the smart home. Subsequently, the setting of interfaces has to be reset to 3 since node 1 holds three interfaces to the RBDS, ZigBee/IEEE 802.15.4 and HomePlug C&C network respectively (for scenarios with two networks, node 1 could be configured to connect both the RBDS and ZigBee/IEEE 802.15.4 network). As with node 0, three interfaces with corresponding parameters are added to node 1 before the node is created. In such way, node 1 is capable of handling packet transfer among the three types of network. In addition, there is no need to repeatedly add another interface 0 to node 1 since it has already been set previously for RBDS network (already coded in our implementation). Any kind of duplicate addition could lead to exceptions during simulations.

Script sample:

```
#configure routing protocol for nodes in a smart home
$ns_ node-config -adhocRouting $rp_(1)      ;#AODV/ZigBee routing

$ns_    controller-config    1                    ;#flag reserved for ZigBee routing:the central controller

#Create node_(1) that is connected to RBDS, 802.15.4 and HomePlugCC network
$ns_ change-numifs    3
#RBDS interface 0 already configured last time
#$ns_ add-ifs 0 $ll_(0) $mac_(0) $ifq_(0) $ifqlen_(0) $ant_(0) $prop_(0) $netif_(0) $chan_(0)
$ns_ add-ifs 1 $ll_(1) $mac_(1) $ifq_(1) $ifqlen_(1) $ant_(1) $prop_(1) $netif_(1) $chan_(1)
$ns_ add-ifs 2 $ll_(2) $mac_(2) $ifq_(2) $ifqlen_(2) $ant_(2) $prop_(2) $netif_(2) $pwr_chan_

set node_(1) [$ns_ node 1]              ;#specify the ID of node as 1
$node_(1) random-motion 0              ;# disable random motion
```

4>Configuration of nodes for the reception of RBDS message

After generation, the controller type of node 1 has to be enabled for later execution. Meanwhile, it is mandatory to confirm how many nodes could be configured to receive incoming RBDS messages and which nodes are enabled to handle those messages. In our implementation, the central controller is capable of sending messages to the remaining nodes in the smart home or only a couple of nodes randomly chosen from the node set.

Script sample:

```
#exclude the first two nodes (node 0 and node 1) from the configuration set
set node_n [expr $val(mn)-2]
$node_(1) NodeType controller enabled      ;#node 1 is set to be the controller
$node_(1) NodeCtrl node_num $node_n    ;#initialization of the number of nodes in a house

#all managed nodes are disabled by default and enable them all
for {set k 2} {$k < $val(mn)} {incr k} {
$node_(1) NodeCtrl rds_pkt enabled $k
}
OR
#node could be configured individually to receive RBDS messages
$node_(1) NodeCtrl rds_pkt enabled 4        ;#node 4 is set to receive RBDS messages
$node_(1) NodeCtrl rds_pkt enabled 6        ;#node 6 is set to receive RBDS messages
```

5>Creation of intermediate nodes with two interfaces

For the remaining nodes in the house, the central controller flag has to be disabled, since only one controller exists among them. Besides, interface 0 is updated to the parameters of the ZigBee/IEEE 802.15.4 network and interface 1 is reset to be configured with the parameters of the HomePlug C&C network.

In addition, the number of nodes could be randomly chosen from the node set as long as it is less than the maximal number of nodes configured in simulations. There is no restriction of how many nodes in the node set could be configured as intermediate nodes, depending on the total number of nodes in the simulations. In other words, any configuration only reflects the demand of scenarios specific to this simulation.

Script sample:
```
#disable the flag for central controller after the creation of central controller
$ns_    controller-config    0

#Create intermediate nodes with both ZigBee/IEEE 802.15.4 and power line homplugC&C interfaces
$ns_    change-numifs    2
$ns_ add-ifs 0 $ll_(1) $mac_(1) $ifq_(1) $ifqlen_(1) $ant_(1) $prop_(1) $netif_(1) $chan_(1)
$ns_ add-ifs 1 $ll_(2) $mac_(2) $ifq_(2) $ifqlen_(2) $ant_(2) $prop_(2) $netif_(2) $pwr_chan_

set node_(2) [$ns_ node 2]        ;#specify the ID of node as 2
$node_(2) random-motion 0         ;#disable random motion

set node_(6) [$ns_ node 6]      ;#specify the ID of node as 6
$node_(6) random-motion 0       ;#disable random motion
… …
```

As for smart home appliances that are plugged onto the power line for data transmission, intermediate nodes could be tailored to the requirement with only the HomePlug C&C interface active in use.

Script sample:
```
#Create intermediate nodes with only the homplugC&C interface active
$ns_    change-numifs    2
$ns_ add-ifs 0 $ll_(1) $mac_(1) $ifq_(1) $ifqlen_(1) $ant_(1) $prop_(1) $netif_(1) $chan_(1)
$ns_ add-ifs 1 $ll_(2) $mac_(2) $ifq_(2) $ifqlen_(2) $ant_(2) $prop_(2) $netif_(2) $pwr_chan_

set node_(7) [$ns_ node 7]        ;#specify the ID of node as 7
$node_(7) random-motion 0         ;#disable random motion

#deactivate the interface to the ZigBee/IEEE 802.15.4 network (on/off)
$node_(7) NodeType ZigIface off
```

6>Creation of normal nodes with one interface to the ZigBee/IEEE 802.15.4 network

For normal nodes in a smart home, only one interface is configured and connected to them afterwards. Since interface 0 has already been configured with parameters for the ZigBee/IEEE 802.15.4 network, it is unnecessary to add a duplicate interface to these nodes for the same reason as node 1.

```
#Create normal nodes attached to ZigBee/IEEE 802.15.4 network
$ns_   change-numifs   1
#ZigBee/IEEE 802.15.4 interface 0 already configured last time
#$ns_ add-ifs 0 $ll_(1) $mac_(1) $ifq_(1) $ifqlen_(1) $ant_(1) $prop_(1) $netif_(1) $chan_(1)

set node_(5) [$ns_ node 5]        ;#specify the ID of node as 5
$node_(5) random-motion 0         ;#disable random motion

set node_(11) [$ns_ node 11]      ;#specify the ID of node as 11
$node_(11) random-motion 0        ;#disable random motion
… …
```

7>Parameters resetting with duplicate propagation models involved

The Two-Ray Ground Model is a slightly improved version of the Free-Space model in that it considers both the direct line-of-sight connection and a ground reflection. The Two-Ray Ground model could present results more accurate than the Free-Space Model if the distance between the transmitter and the receiver is long. As compared to the Free-Space model, the Two-Ray Ground Model is more suitable for most short-distance wireless communications. The Shadowing model emphasizes fading effects for signals due to multipath diffusion in the case of wireless communications at a considerable distance. As a matter of fact, both the Two-Ray Ground model and the Shadowing model could be adopted in the situation of smart homes with a limited distance in between.

In the NS-2 environment, the parameters of the propagation model could be assigned statically through the simulation script and then be passed through the NS-2 simulator object to the corresponding model object in C++ space.

Script sample:

```
Propagation/Shadowing set pathlossExp_ 1.7
Propagation/Shadowing set std_db_ 7.0
Propagation/Shadowing set dist0_ 1.0
Propagation/Shadowing set seed_ 0
```

However, the central controller and the intermediate nodes may experience a situation where duplicate models for the ZigBee/IEEE 802.15.4 network and the HomePlug C&C network involved in both networks share the same propagation model with slightly different parameter values. Besides, the NS-2 execution environment provides no support for the duplicate assignment of one model. In this case, the first static assignment is replaced by the second static assignment of the same model before nodes are generated. To avoid such a problem, each instance of the propagation model corresponding to the interface attached to nodes has to be obtained from the NS-2 simulator object so as to reset the values for all nodes simulating the smart house.

Script sample:

```
#parameter resetting of propagation model for ZigBee/IEEE 802.15.4 network
if { $prop_(1) == "Propagation/Shadowing"} {
    set zig_prop_inst [$ns_ set propInstance_(0)]

    $zig_prop_inst set pathlossExp_ 1.7
    $zig_prop_inst set std_db_ 7.0
    $zig_prop_inst set dist0_ 1.0
    $zig_prop_inst set seed_ 0
}

#parameter resetting of propagation model for HomePlug C&C network
if { $prop_(2) == "Propagation/Shadowing"} {
    set hmplg_prop_inst [$ns_ set propInstance_(1)]

    $hmplg_prop_inst set pathlossExp_ 2.0
    $hmplg_prop_inst set std_db_ 4.0
    $hmplg_prop_inst set dist0_ 1.0
    $hmplg_prop_inst set seed_ 0
}
```

8>Energy model setting for nodes in smart homes

In most cases, sensor-enabled nodes with limited energy in a smart home are mostly battery powered. Thus, it is reasonable to explore the dynamics of energy dissipation for these nodes in simulations. In the NS-2 environment, an energy model (only one generic model defined in NS-2) used to update energy consumption of a node and an initial value of energy are declared and connected to nodes through node configuration and subsequent generation.

Script sample:

```
#Energy model type and initialization of energy value
set val(energymodel)     EnergyModel                  ;# Energy Model Type
set val(initialenergy)   100                          ;# initial value

#set energy parameters when configuring the controller
$ns_ node-config -adhocRouting $rp_(1) \
                 -energyModel $val(energymodel) \
                 -initialEnergy $val(initialenergy)
```

Considering the fact that energy consumption is only calculated by the PHY layer object, the assignment of transmission and reception power has to be done to each interface targeted for the specific PHY layer in a node with multiple interfaces.

Script sample:

```
#set the power of transmission and reception for the controller to each interface
$node_(1) setPhyPt 0    30.32e-3                ;#value is randomly chosen for generic tests
$node_(1) setPhyPr 0    34.28e-3

$node_(1) setPhyPt 1    31.32e-3
$node_(1) setPhyPr 1    35.28e-3

$node_(1) setPhyPt 2    30.32e-3
$node_(1) setPhyPr 2    34.28e-3
```

```
#set the power of transmission and reception for intermediate nodes to each interface
$node_(2) setPhyPt 0    31.32e-3
$node_(2) setPhyPr 0    35.28e-3

$node_(2) setPhyPt 1    30.32e-3
$node_(2) setPhyPr 1    34.28e-3

#set the power of transmission and reception for normal nodes to its interface
$node_(5) setPhyPt 0    31.32e-3
$node_(5) setPhyPr 0    35.28e-3
```

9>Error model setting for nodes in smart homes

To support the reliability-based routing in simulations, each node in a smart home is equipped with error models, depending upon the amount of interface attached to the node. In our implementation, the unit of error is identified as packet, while the random variable used for generating packet errors is uniformly distributed from 0 to 1. Meanwhile, the packet error rate of the error model could be adjusted accordingly to cope with various scenarios. Additionally, it is assumed that each error object generated by a single procedure should be added to the target node through an interface-oriented command prior to the node construction, as exemplified as follows:

Script sample:

```
proc UniformErr1 {} {                        ;# procedure for generating error model
set err [new ErrorModel]                     ;# error object
$err set rate_ 0.05                          ;# the packet error rate of 5 percent
$err unit pkt                                ;# the unit of error (packet, byte and bit)
$err ranvar [new RandomVariable/Uniform] ;# random variable uniformly distributed from 0 to 1
$err drop-target [new Agent/Null]
return $err
}

proc UniformErr2 {} {
set err [new ErrorModel]
$err set rate_ 0.01
$err unit pkt
$err ranvar [new RandomVariable/Uniform]
$err drop-target [new Agent/Null]
return $err
}

#Create nodes with both ZigBee/IEEE 802.15.4 and HomePlug C&C interfaces
$ns_    change-numifs    2              ;# mobile node equipped with two interfaces
$ns_  add-ifs-err  0  $ll_(1)  $mac_(1)  $ifq_(1)  $ifqlen_(1)  $ant_(1)  $prop_(1)  $netif_(1)  $chan_(1)
UniformErr1                             ;# bind an error object with the corresponding interface
$ns_  add-ifs-err  1  $ll_(2)  $mac_(2)  $ifq_(2)  $ifqlen_(2)  $ant_(2)  $prop_(2)  $netif_(2)  $pwr_chan_
UniformErr2
```

10>General setting of the queue model in simulations

The Queue Model specifies the queuing mechanism in NS-2 simulations, where packets are pushed to or removed from the queue when they are ready for transmission.

As a common queue type mostly used in simulations, Drop-Tail implements the FIFO (First-In-First-Out) queuing mechanism where packets entering the queue will be handled first. Meanwhile, the last packet entering the queue will be discarded once the queue reaches its limit pre-defined in the scenario script file. In addition, the queue length has no noticeable impact upon the network performance and only depends on the efficient utilization of network resources available during executions. Thus, there is no technical restriction of how long in size it can be set in simulations.

Script sample:

```
set ifq_(1)              Queue/DropTail/PriQueue       ;# interface queue type
set ifqlen_(1)           50                            ;# max packet in ifq
```

11>Setting of power parameters for PHY layer

Transmission Power (Pt_) is defined as the power with which the signal is transmitted. ZigBee/IEEE 802.15.4-based devices could operate at a transmit power of 0dBm (1mW) regulated by the ZigBee/IEEE 802.15.4 specification [21]. Thus, Pt_ could be set to 0dBm (1mW) given an ideal case in simulations.

Reception power Threshold (RXThresh_) identifies the least power level of packets to be detected successfully in data transmission. The packets are dropped if the received signal power is lower than the current threshold pre-defined for nodes.

Carrier Sense Threshold (CSThresh_) specifies the sensing range of nodes in simulations. Data transmission can be sensed as long as the received signal strength is higher than the current threshold. Besides, a successful detection of packets depends on whether the nodes are capable of sensing a range at least equal to RXThresh_ in terms of threshold.

Given the energy model specified in nodes, Transmit Power (txPower), Receive Power (rxPower) and Idle Power (idlePower) are correspondingly assigned to each PHY layer of nodes for the purpose of calculating energy consumption in different transmission states.

txPower identifies the power consumed when transmitting a data packet; rxPower identifies the power consumed by nodes to receive a data packet; idlePower identifies the power consumed by the node during the idle/sleep mode. It is to be noted that the ZigBee/IEEE 802.15.4-based devices are normally designed to operate in idle/sleep mode most of the time and are woken up to handle packets occasionally in realistic environment. Thus, it is optional for nodes to be configured with such mode in simulations, depending upon the requirements specific to the simulation scenarios.

Script sample for the central contrller:

```
$node_(1) setPhyPt 0    30.32e-3
$node_(1) setPhyPr 0    34.28e-3

$node_(1) setPhyPt 1    31.32e-3
$node_(1) setPhyPr 1    35.28e-3

$node_(1) setPhyPt 2    30.32e-3
```

$node_(1) setPhyPr 2   34.28e-3

These parameters at the PHY layer mentioned above are determined by the capabilities of realistic devices including the signal power of transceiver, electric current and voltage supported by devices, etc. Hence, these parameters are supposed to be configured on the basis of realistic environments with devices available for on-site experiments. In the normal case, the script segments of the RBDS network in [10], the ZigBee/IEEE 802.15.4 network with ZigBee routing in [30] as well as the examples of the ZigBee/IEEE 802.15.4 network with AODV routing included at /ns2-allinone-2.33/NS-2.33/ex/wpan could be combined to represent generic simulation scenarios.

10.4 Parameter Configuration for Different Networks

10.4.1 RBDS

A script in generic simulations mostly contains parameter configurations for a specific network, including settings of the routing protocol layer, LL, MAC/PHY layer, the queue type for LL/MAC and the maximal length of the queue involved in buffering outgoing packets, an antenna model as well as a propagation model adopted for nodes. In the RBDS network, the emphasis is placed on the parameter settings of the MAC/PHY layer and the propagation model intended for issues of burst errors due to large scale fading since it addresses the data communication at a considerable distance. The section on the RBDS network configuration refers to the annotation explained by the author in Chapter 5 and Appendix C in [10] and the configuration of parameters at the PHY layer is summarized in Table 3. There is no special requirement in terms of smart home networking simulations since our major focus is placed inside the house.

Transmitter Power	27kW
Transmitter Height	210m
Receiver Height	1.5m
Receiver Sensitivity (Threshold)	-103dBm (-133dB); RSSI = 0
Path-loss Exponent	3.50
Shadowing Deviation	12
Ricean K Value	0.0
Ricean max Velocity	120 km/h

Table 3 PHY layer parameters based after calibration [10]

Script sample:

```
set rp_(0)          DumbAgent         ;# routing protocol for one-hop routing
set   ll_(0)        LL                ;# link layer type

# interface queue type (unnecessary for RBDS, leave empty)
set ifq_(0)         Queue/DropTail/PriQueue
set ifqlen_(0)      50                ;# max packet in ifq

set   mac_(0)       Mac/RDS           ;# MAC type

# set the bandwidth for the MAC
Mac/RDS set bandwidth_ 1187.5   ;# RBDS bitrate
Mac/RDS set mode_ 1             ;# mode for RBDS 0= Normal Mode ; 1 = Real-Time mode
Mac/RDS set NTX_repeat_ 5       ;# Number of times a message is repeated
Mac/RDS set DTX_new_ 10

set netif_(0)      Phy/WirelessPhy        ;# network interface type

# set the carrier frequency
Phy/WirelessPhy set freq_ 89.9e+6

#The power levels of the transmitters and recievers
# transmit power
Phy/WirelessPhy set Pt_ 27.00e3

# Receive sensitivity.
```

```
Phy/WirelessPhy set RXThresh_ 5.011872336e-14
Phy/WirelessPhy set CSThresh_ 5.011872336e-14

set ant_(0)          Antenna/OmniAntenna        ;# antenna model

# Rayleigh and Ricean with Shadowing as the large scale fading (Propagation model for wireless
channel)
set prop_(0)         Propagation/RiceanShadowing

# Parameters that characterize the large scale fading
Propagation/RiceanShadowing set pathlossExp_ 3.5
Propagation/RiceanShadowing set std_db_ 12
Propagation/RiceanShadowing set dist0_ 45.125
Propagation/RiceanShadowing set seed_ [expr {int(rand()*64)}] ; # seed for the power deviation
random var
; # to obtain identical simulation fix this to a number between 0 and 63
; # (which indexes one of the 64 pre-defined seeds in NS-2 ) / uncomment the line below
# Propagation/RiceanShadowing set seed_ 0

if {$opt(seed) > 0} {
    puts "Seeding Random number generator with $opt(seed)\n"
    ns-random $opt(seed)
}

set val(RiceanK)          0.0     ;# Ricean K factor
set val(RiceanMaxVel)     120     ;# Ricean   Propagation   MaxVelocity Parameter

# number of Nodes in the RBDS network   (Node 0 is the base station)
set val(nn)    2

# Ricean   Propagation: Maximum ID of nodes (Total number of nodes) used to
# compute pairwise table offsets.
set val(RiceMaxNodeID)   [expr {$val(nn)-1}]
set val(RiceDataFile)    rice_table.txt            ;# Ricean Propagation Data File

# propagation settings after the node is generated
```

```
if { $prop_(0) == "Propagation/RiceanShadowing"} {
        set prop_inst [$ns_ set propInstance_(0)]

        $prop_inst MaxVelocity    $val(RiceanMaxVel);
        $prop_inst RiceanK          $val(RiceanK);
        $prop_inst LoadRiceFile    $val(RiceDataFile);
        $prop_inst RiceMaxNodeID $val(RiceMaxNodeID);
}
```

10.4.2 ZigBee/IEEE 802.15.4

The parameter configuration of ZigBee/IEEE 802.15.4 for AODV routing refers to the examples provided by the authors of the ZigBee/IEEE 802.15.4 MAC/PHY layer implementation in /ns2-allinone-2.33/NS-2.33/ex/wpan. There are no special parameters to be adjusted in terms of generic tests in smart home networking scenarios.

Script sample for AODV routing:

```
set prop_(1)          Propagation/TwoRayGround     ;# radio-propagation model
set netif_(1)         Phy/WirelessPhy/802_15_4     ;# PHY type
set mac_(1)           Mac/802_15_4                 ;#MAC type
set ifq_(1)           Queue/DropTail/PriQueue      ;# interface queue type
set ifqlen_(1)        50                           ;# max packet in ifq
set ll_(1)            LL                           ;# link layer type
set ant_(1)           Antenna/OmniAntenna          ;# antenna model
set rp_(1)            AODV                         ;# AODV routing protocol

Mac/802_15_4 wpanCmd verbose on

# For model 'TwoRayGround'
set dist(12m) 1.33527e-06

Phy/WirelessPhy/802_15_4 set CSThresh_ $dist(12m)
Phy/WirelessPhy/802_15_4 set RXThresh_ $dist(12m)

#change back to default value
```

```
Phy/WirelessPhy/802_15_4 set Pt_ 0.28183815
Phy/WirelessPhy/802_15_4 set freq_ 914e+6
```

The parameter configuration of ZigBee/IEEE 802.15.4 for ZigBee routing refers to the script provided by the author of the ZigBee routing implementation in [30]. In this script, the author adopted the shadowing model instead of the two-way ground model in the simulation. Meanwhile, the values of the path loss exponent and power deviation employed in the shadowing model are obtained empirically by field measurements rather than equation calculation. Due to the lack of support of on-site experiments based on such model, it is reasonable to directly use the parameters of propagation model and PHY layer presented by the author for generic tests in smart homes.

Script sample for ZigBee routing:

```
set prop_(1)          Propagation/Shadowing        ;# radio-propagation model
set netif_(1)         Phy/WirelessPhy/802_15_4     ;#PHY type
set mac_(1)           Mac/802_15_4                 ;#MAC type
set ifq_(1)           Queue/DropTail/PriQueue      ;# interface queue type
set ifqlen_(1)        50                           ;#max packet in ifq
set ll_(1)            LL                           ;#link layer type
set ant_(1)           Antenna/OmniAntenna          ;#antenna model
set rp_(1)            Zigbee                       ;#ZigBee routing protocol

Propagation/Shadowing set pathlossExp_ 1.7
Propagation/Shadowing set std_db_ 7.0
Propagation/Shadowing set dist0_ 1.0
Propagation/Shadowing set seed_ 0

Phy/WirelessPhy/802_15_4 set CSThresh_ 1.0e-12
Phy/WirelessPhy/802_15_4 set RXThresh_ 1.0e-12
Phy/WirelessPhy/802_15_4 set Pt_ 0.01
Phy/WirelessPhy/802_15_4 set freq_ 2.4e+9          ;#operates in the 2.4 GHz band
```

In the setting of a ZigBee/IEEE 802.15.4 network, node type configuration is closely tied to device association and beacon transmission [31]. As explained in the section on

ZigBee/IEEE 802.15.4 networks in Chapter 6, device association emphasizes the formation of a cluster tree over the underlying communication link while beacon transmission or beacon-enabled transmission may result in higher overhead and delay in networks. Therefore, node type configuration for AODV routing is optional and flexible (no fundamental influence in terms of data transmission in simulations, depending on the location of nodes and the corresponding transmission and reception power).

As compared to AODV routing, the author of ZigBee routing forced all devices to be connected into the network by locating any sub-root around it before data transmission (the central controller called coordinator is configured to be the root of the cluster tree). Any type of failure to associate with the network will lead to inaccessibility of the destination node and in turn impact the whole network traffic. To be fair to both routing approaches in terms of metric comparison, node type configuration should also be enabled for AODV routing. Taking the time spent on device association into account, it is reasonable to enable data transmission to be executed after all devices have successfully connected to the network through association.

Script sample of the setting of node type for AODV routing:

```
$ns_ at 0.0     "$node_(1) NodeLabel PAN Coor"
$ns_ at 0.0     "$node_(1) sscs startPANCoord 1"          ;#     startPANCoord     <txBeacon=1>
<BO=3> <SO=3>
$ns_ at 0.5     "$node_(2) sscs startDevice 1 1 1"        ;#     startDevice       <isFFD=1>
<assoPermit=1> <txBeacon=0> <BO=3> <SO=3>
… …
$ns_ at 5.5     "$node_(7) sscs startDevice 0"
… …
```

Script sample of the setting of node type for ZigBee routing:

```
# startPANCoord <txBeacon=1> <BO=3> <SO=3>
$ns_ at 0.0 "$node_(1) sscs startPANCoord 0 15"          ;#in non-beacon mode

set start_time_(2) 108.860000
$ns_ at 1.0 $start_time_(2) "$node_(2) sscs startDevice 1 1 0 15"
```

… …

Normally, the group of parameters associated with ZigBee routing should be kept unchanged for proper execution in simulations.

First of all, buffLimit, buffer_pool, and drop_front_ are related to the data and command buffer for data forwarding in ZigBee routing, while nwkMaxDepth represents the maximal addressing depth for data routing.

Secondly, nwkMaxRouters, nwkMaxChildren, pkt_consume_delay are used for device association in that it is mandatory for a ZigBee device to be connected in the ZigBee network (cluster tree structure) before data transmission.

Hence, there is no need to make any adjustment to the parameters mentioned above by default for generic tests in smart home networking simulations.

Script sample intended for ZigBee routing:

```
#limitation of data and command packet buffer in transmission, -1 means no limit
$ns_ at 0.0 “$node_(1) sscs buffLimit -1”

#the size of buffer pool
$ns_ at 0.0 “$node_(1) sscs buffer_pool 2500000”

#drop the front packets when data or command buffer is full
Agent/Zigbee set drop_front_ 1

#setting of maximal depth for routing in terms of radius
$ns_ at 0.0 “$node_(1) sscs nwkMaxDepth 20”

#setting for device association
#the maximal number of routers that can associate with this device
$ns_ at 0.0 “$node_(1) sscs nwkMaxRouters 200”;$ns_ at 0.0 “
```

```
#the max number of children that can associate with this device
$node_(1) sscs nwkMaxChildren 200"

#packet delay for the reception of association request/response during device association
$ns_ at 0.0 "$node_(1) sscs pkt_consume_delay 1"
```

The current implementation of ZigBee routing partially supports tree routing in the sense that all packets are first forwarded to the root of the network tree before reaching the next hop. Considering that there is no use in our project for this type of routing, nwkUseTreeRouting is set to 0 in the script file.

```
#flag of tree routing in ZigBee/IEEE 802.15.4 network
$ns_ at 0.0 "$node_(1) sscs nwkUseTreeRouting 0"
```

In order to keep backwards compatibility with the existing implementation of the ZigBee/IEEE 802.15.4 MAC/PHY layer in NS-2, a flag ZigBeeRouting is defined in our project to enable ZigBee routing when this type of routing is employed in simulations in case any unexpected collision with AODV routing occurs during execution. For AODV routing, the flag should be disabled by default.

```
#back compatibility with existing implementation in NS-2
Mac/802_15_4 wpanCmd ZigbeeRouting on        ;# on/off for ZigBee routing
```

To configure a Flooding protocol in a smart home, the protocol name previously specified in simulations is replaced with "RdsFlooding" defined in our implementation except for the setting of underlying networks identical to that of AODV/ZigBee routing. Meanwhile, all other attributes associated with AODV/ZigBee routing must be removed in advance or commented out with "#" in the scenario scripts before the creation of nodes, since the Flooding protocol deployed in simulations is independent from AODV/ZigBee routing.

Script sample for Flooding protocol:

```
set prop_(1)                Propagation/TwoRayGround      ;# radio-propagation model
set netif_(1)            Phy/WirelessPhy/802_15_4       ;# PHY type
set mac_(1)              Mac/802_15_4                   ;#MAC type
set ifq_(1)              Queue/DropTail/PriQueue        ;# interface queue type
set ifqlen_(1)           50                             ;# max packet in ifq
set ll_(1)               LL                             ;# link layer type
set ant_(1)              Antenna/OmniAntenna            ;# antenna model
set rp_(1)               RdsFlooding           ;# Flooding protocol
set val(mn)              22                    ;# number of mobile nodes in the whole network

$node_(1) NodeType controller enabled

#set node_n [expr $val(mn)-2]
#$node_(1) NodeCtrl node_num $node_n           ;#reserved for AODV/ZigBee routing

#for {set k 2} {$k < $val(mn)} {incr k} {      ;#reserved for AODV/ZigBee routing
#$node_(1) NodeCtrl rds_pkt enabled $k
#}

#Mac/802_15_4 wpanCmd ZigbeeRouting on        ;#flag reserved for ZigBee routing
#$ns_   controller-config   1                 ;#reserved for ZigBee routing: the central controller
#$ns_   controller-config   0                 ;#reserved for ZigBee routing: the remaining nodes
```

10.4.3 HomePlug C&C

Considering that the HomePlug C&C protocol stacks is a clone of the IEEE 802.11 wireless LAN MAC/PHY stack included in NS-2 v2.33, all parameters involved could flexibly be configured based on examples available for IEEE 802.11 in /ns2-allinone-2.33/NS-2.33/ex since IEEE 802.11 is wildly used in most simulations associated with wireless short-range communications. Specifically, the data rate of the cloned implementation is reduced to 25Kbps, which is significantly lower than that of ZigBee/IEEE 802.15.4 (250Kbps at 2.4 GHz), so as to represent an approximation to data transmission at a speed of 7.5 Kbps in the HomePlug C&C network. Further reductions of the data rate, as discussed earlier, are not possible, due to the limitation of NS-2 simulation environment as well as the complexity of event scheduling in the cloned implementation.

Script sample:

```
#Define options for HomePlugC&C PHY/MAC layers
set val(pwrchan)        Channel/PowerChannel        ;# Power line Channel Type
set prop_(2)            Propagation/Shadowing       ;# radio-propagation model
set netif_(2)           Phy/HomePlugCCPhy           ;# PHY type
set mac_(2)             Mac/HomePlugCC              ;#MAC type
set ifq_(2)             Queue/DropTail/PriQueue     ;# interface queue type
set ifqlen_(2)          50                          ;# max packet in ifq
set ll_(2)              LL                          ;# link layer type
set ant_(2)             Antenna/OmniAntenna         ;# antenna model

Mac/HomePlugCC set basicRate_    25Kb      ;# set this to 0 if want to use bandwidth
Mac/HomePlugCC set dataRate_    25Kb    ;# both control and data packets
Mac/HomeplugCC set PLCPDataRate_      25000              ;# 25Kbps at the PHY Layer (PLCP)
Mac/HomeplugCC set bandwidth_    25000    ;#unused if basicRate and dataRate are preconfigured
```

10.5 Configuration for Data Traffic among Networks

In most cases, two types of data communications are involved in our project: short RBDS messages issued by the RBDS sender to the central controller, which forwards the reframed messages to nodes preconfigured, and packets of short size transmitted from nodes in a house to the central controller. Hence, data transmissions based on UDP is used in simulations. Besides, the port numbered 0 is used by all nodes for RBDS message transmission since it is hardcoded in our implementation. Consequently, data transmission in our networks could use any port number starting from 1. In this case, the communication of RBDS messages among nodes has to be configured in the first place so as to occupy the port number 0.

RBDS message transmission refers to the annotation explained in the application layer part of Appendix C in [10]. An application of Title-24 based on UDP transport layer is connected to the RBDS sender. The rest of the nodes including the central controller are configured as sink nodes for the reception of RBDS messages. Meanwhile, the RBDS communication link is only established between the RBDS sender and the central controller since nodes in a house receive the RBDS messages only through the central controller.

Script sample:

```
#Create a UDP agent and attach it to node 0
set RDSudp [new Agent/UDP]
$ns_ attach-agent $node_(0) $RDSudp
$RDSudp set dst_addr_ 0

#Create a Title-24 application here
set tt24 [new Application/Title-24]
$tt24 set interval_ 100.0                    ;#value is randomly chosen for generic tests
$tt24 set size_ 30
$tt24 attach-agent $RDSudp

#Create sink node for reception of RBDS messages
for {set j 1} {$j < $val(mn)} {incr j} {
        set RDSnull_($j) [new Agent/Null]
        $ns_ attach-agent $node_($j) $RDSnull_($j)
}

#Connect the RBDS sender and the central controller
$ns_ connect $RDSudp     $RDSnull_(1)

;#RBDS message transfer is set to the right time after all devices have connected to the network in smart homes, depending on the time of device association..

$ns_ at 700.0 "$tt24 start"
```

As with data transmission based on the UDP layer in most simulations, packets are generated by a CBR traffic generator from intermediate nodes or normal nodes to the central controller. There are no special requirements for generic tests except that the size of the data packet and the interval of transmission should remain reasonable to smart home networking simulations.

Script sample:

```
proc cbrtraffic { interval src dst stime } {
    global ns_ node_
    set udp_($src) [new Agent/UDP]
    eval $ns_ attach-agent \$node_($src) \$udp_($src)
    set null_($dst) [new Agent/Null]
    eval $ns_ attach-agent \$node_($dst) \$null_($dst)
    set cbr_($src) [new Application/Traffic/CBR]
    eval \$cbr_($src) set packetSize_ 70                    ; # 70 bytes in size
    eval \$cbr_($src) set interval_ $interval                ;#100 seconds
    eval \$cbr_($src) set random_ 1
    eval \$cbr_($src) attach-agent \$udp_($src)
    eval $ns_ connect \$udp_($src) \$null_($dst)

    $ns_ at $stime "$cbr_($src) start"
}

set appTime1         550.0      ;# in seconds
set val(traffic)      cbr       ;# network traffic type

if { ("$val(traffic)" == "cbr") } {
    puts "\nTraffic: $val(traffic)"

    $ns_ at $appTime1 "$ns_ trace-annotate \"(at $appTime1) cbr traffic from node 10 to node 1\""
    cbrtraffic 100 10 1    $appTime1    ;#UDP packet sent from node 10 to 1 every 100 seconds

}
```

10.6 Configuration of Routing Strategy

Preset with the AODV or ZigBee routing protocol, the combination of wired and wireless networks simulating the traffic in a smart home operates with distinct routing strategies indentified in the scenario scripts. By default, all simulations run with the joint path routing option (discussed in Section 7.3). Reliability-based routing is performed with error models configured for each node with multiple interfaces in a smart home (detailed in Sections 4.3.3 and 10.1). Besides, the steps of configuring extra routing preference are detailed as follows:

1>Routing via the ZigBee/IEEE 802.15.4 Networks

With regard to the routing option via only the ZigBee/IEEE 802.15.4 network, the HomePlug C&C interface attached to the corresponding nodes is disabled in the scenario scripts in that no traffic occurs over the unused network. Under such circumstances, the amount of interfaces intended for the central controller is reduced to 2 (RBDS and ZigBee/IEEE 802.15.4), whereas the remaining nodes are equipped with only one interface (ZigBee/IEEE 802.15.4).

Script sample:

```
set val(ni)                 2               ;#total number of interface/channel

#Create the central controller with the RBDS and ZigBee/IEEE 802.15.4 interface (HomPlugC&C disabled)
$ns_   change-numifs   2
#$ns_ add-ifs 0 $ll_(0) $mac_(0) $ifq_(0) $ifqlen_(0) $ant_(0) $prop_(0) $netif_(0) $chan_(0)
$ns_ add-ifs 1 $ll_(1) $mac_(1) $ifq_(1) $ifqlen_(1) $ant_(1) $prop_(1) $netif_(1) $chan_(1)
#$ns_ add-ifs 2 $ll_(2) $mac_(2) $ifq_(2) $ifqlen_(2) $ant_(2) $prop_(2) $netif_(2) $pwr_chan_

#Create nodes with the ZigBee/IEEE 802.15.4 interface (HomPlugC&C disabled)
$ns_   change-numifs   1
$ns_ add-ifs 0 $ll_(1) $mac_(1) $ifq_(1) $ifqlen_(1) $ant_(1) $prop_(1) $netif_(1) $chan_(1)
#$ns_ add-ifs 1 $ll_(2) $mac_(2) $ifq_(2) $ifqlen_(2) $ant_(2) $prop_(2) $netif_(2) $pwr_chan_

#Create nodes only with ZigBee/IEEE 802.15.4 connection
$ns_   change-numifs   1
# ZigBee/IEEE 802.15.4 interface 0 already configured in NS-2
#$ns_ add-ifs 0 $ll_(1) $mac_(1) $ifq_(1) $ifqlen_(1) $ant_(1) $prop_(1) $netif_(1) $chan_(1)
```

2>Backbone-Based Routing and Dual Routing

A static attribute is created separately for backbone-based routing and dual routing in our implementation, which enables the nodes in a smart home to share the same routing options when transmitting packets. To identify the routing option, the corresponding attribute is switched on prior to simulations except for the basic configuration for multiple networks. It is to be noted that both two routing options are deployed independently from one another according to the specific scenarios. Coexistence of them in the same scenario scripts will give rise to unexpected errors in our simulation model.

Script sample:

```
$node_(1) NodeType EnergyPrio on          ;# Backbone-Based Routing (on/off)
OR
$node_(1) NodeType DualRoute on           ;# Dual Routing (on/off)
```

References

[1] "Smart Grid: Enabler of the New Energy Economy", December 2008, [Online]. Available: http://www.oe.energy.gov/DocumentsandMedia/final-smart-grid-report.pdf

[2] Federal Energy Regulatory Commission, "Staff Report-Assessment of Demand Response and Advanced Metering", September 2007, [Online]. Available: http://www.ferc.gov/legal/staff-reports/09-07-demand-response.pdf

[3] Peak Load Management Alliance, "Interaction of Advanced Metering Initiative (AMI) and Demand Response", Spring 2006 Conference, March 13, 2006, [Online]. Available: http://www.peaklma.com/documents/Haynes.pdf

[4] Joshua Horton, "AMI as Demand Response Enabling Technology", Automation Insight, February 2009, pp. 1-2, [Online]. Available: http://kema.fr/cn/Images/Automation%20Insight%20-screen%202-09.pdf

[5] Ontario's Smart Grid Forum, "Enabling Tomorrow's Electricity System: Report of the Ontario Smart Grid Forum", February 2009, [Online]. Available: http://www.ieso.ca/imoweb/pubs/smart_grid/Smart_Grid_Forum-Report.pdf

[6] Nunes, R., Delgado, J., "An architecture for a home automation system", IEEE International Conference on Electronics, Circuits and Systems, Volume 1, Sept. 1998, pp. 259-262.

[7] Ricquebourg, V., Menga, D., Durand, D., Marhic, B., Delahoche, L., Loge, C., "The Smart Home Concept: our immediate future", 1ST IEEE International Conference on E-Learning in Industrial Electronics, Dec. 2006, pp. 23-28.

[8] M. Daucher, E. Gartner, M. Cortler, W. Keller, and H. Kuhr, "RDS-Radio Data System: A Challenge and a Solution", Fujitsu Ten Technical Journal, 30, November 2008.

[9] Patti Harper-Slaboszewicz, "FM Radio Offers Intriguing Demand Response Solution", Daily IssueAlert, November 11, 2005, [Online]. Available: http://www.e-radioinc.com/umc.htm

[10] Monageng Kgwadi, "Communication Protocol for Residential Electrical Demand Response in Home Devices", Master thesis, Carleton University, July 2009

[11] Yousuf, M.S., El-Shafei, M., "Power Line Communications: An Overview - Part I", 4th International Conference on Innovations in Information Technology, November 2007, pp. 218-222.

[12] The Network Simulator ns-2 Manual,
[Online]. Available: http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf

[13] Teerawat Issariyakul, Ekram Hossain, "Introduction to Network Simulator NS2", Springer Science+Business Media, LLC, New York, USA, 2009

[14] The Enhanced Network Simulator(TeNs),
[Online]. Available: http://www.cse.iitk.ac.in/users/braman/tens

[15] Tzi-cker Chiueh, Ashish Raniwala, "Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network", In Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies, March 2005, vol.3, pp. 2223-2234.

[16] Ramon Aguero Calvo, "Adding Multiple Interface Support in NS-2", January 2007,
[Online]. Available:
http://personales.unican.es/aguerocr/files/ucMultiIfacesSupport.pdf

[17] Laurent Paquereau, Bjarne E. Helvik, "A module-based wireless node for NS-2", In Proceeding from the 2006 workshop on ns-2: the IP network simulator, 2006, Vol.202, Article No.4.

[18] Laurent Paquereau,"Extensions to ns-2", October 29, 2009,
[Online]. Available: http://people.item.ntnu.no/~paquerea/ns/q2s_doc.pdf

[19] Nicola Baldo, Federico Maguolo, Marco Miozzo, Michele Rossi, Michele Zorzi, "ns2-MIRACLE: a modular framework for multi-technology and cross-layer support in network simulator 2", In Proceedings of the 2nd international conference on Performance evaluation methodologies and tools, 2007, Vol.321, Article No.16.

[20] NS-MIRACLE: Multi-InteRfAce Cross-Layer Extension library for the Network Simulator, [Online]. Available:
http://www.dei.unipd.it/wdyn/?sez_alias=ricerca/signet/tools/nsmiracle

[21] Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs), June 2006, [Online]. Available:
http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf

[22] Marina Petrova, Janne Riihijarvi, Petri Mahonen, Saverio Labella, "Performance study of IEEE 802.15.4 using measurements and simulations", In IEEE Wireless Communications and Networking Conference, April 2006,Vol.1, pp. 487-492.

[23] Yu-Ju Lin, Latchman, H.A., Minkyu Lee, Katar, S., "A power line communication network infrastructure for the smart home", IEEE Wireless Communications, Dec. 2002, Volume 9, Issue 6, pp. 104-111.

[24] Min-Soo Kim, Dong-Min Son, Young-Bae Ko, Young-Hyun Kim, "A Simulation Study of the PLC-MAC Performance using Network Simulator-2", IEEE International Symposium on Power Line Communications and Its Applications, April 2008, pp. 99-104

[25] ZigBee Alliance, "ZigBee Specification", June 27, 2005, [Online]. Available: http://www.zigbee.org/Markets/ZigBeeSmartEnergy/Specification.aspx

[26] Runcai Huang, Yiwen Zhuang, Qiying Cao, "Simulation and Analysis of MFlood Protocol in Wireless Network", International Symposium on Computer Science and Computational Technology, Dec. 2008, Vol.1, pp. 658-662.

[27] J. Zheng and M. J. Lee, "A comprehensive performance study of IEEE 802.15.4", Sensor Network Operations, IEEE Press, Wiley Interscience, 2006, Chapter 4, pp. 218-237, [Online]. Available: http://www-ee.ccny.cuny.edu/zheng/papers/paper1_wpan_performance.pdf

[28] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, Jang-Ping Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network",Wireless Networks, March-May 2002, Volume 8, Issue 2/3, pp. 153-167.

[29] Pradeep Kyasanur, Nitin H. Vaidya, "Routing and Interface Assignment in Multi-ChannelMulti-Interface Wireless Networks", IEEE Wireless Communications and Networking Conference, March 2005, Vol.4, pp. 2051-2056.

[30] ZigBee/IEEE 802.15.4 Module for NS2 simulator, Oct. 19, 2008, [Online]. Available: http://www.cs.uwm.edu/~mukul/wpan.html

[31] Jianliang Zheng, "802.15.4 and ZigBee Routing Simulation", 2003, [Online]. Available: http://www-ee.ccny.cuny.edu/zheng/pub/file/WPAN_ZBR_pub.pdf