DESIGN OF REAL-TIME SIMULATION TESTBED FOR ADVANCED METERING INFRASTRUCTURE (AMI) NETWORK

A Thesis

by

ABHIJEET SAHU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Karen L. Butler-Purry
Co-Chair of Committee,	Ana Elisa Goulart
Committee Members,	Pierce Cantrell, Jr
	Le Xie
Head of Department,	Miroslav M. Begovic

May 2018

Major Subject: Electrical Engineering

ABSTRACT

Conventional power grids are being superseded by smart grids, which have smart meters as one of the key components. Currently, for the smart metering communication, wireless technologies have predominantly replaced the traditional Power Line Communication (PLC). Different vendors manufacture smart meters using different wireless communication technologies. For example, some vendors use WiMAX, others prefer Low-Power Wireless Personal Area Networks (Lo-WPAN) for the Media Access Control (MAC) and physical layer of the smart meter network, also known as Advanced Metering Infrastructure (AMI) network. Different communication techniques are used in various components of an AMI network. Thus, it is essential to create a testbed to evaluate the performance of a new wireless technology or a novel protocol to the network. It is risky to study cyber-security threats in an operational network. Hence, a real-time simulation testbed is considered as a substitute to capture communication among cyber-physical subsystems.

To design the communication part of our testbed, we explored a Cellular Internet of Things (C-IoT) : Co-operative Ultra NarrowBand (C-UNB) technology for the physical and the MAC layer of the Neighborhood Area Network (NAN) of the AMI. After successful evaluation of its performance in a Simpy python simulator, we integrated a module into Network Simulator-3 (NS-3). As NS-3 provides a platform to incorporate real-time traffic to the AMI network, we can inject traffic from power simulators like Real Time Digital Simulator (RTDS). Our testbed was used to make a comparative study of different wireless technologies such as IEEE 802.11ah, WiMAX, and Long Term Evolution (LTE). For the traffic, we used HTTP and Constrained Application Protocol (CoAP), a widely used protocol in IoT. Additionally, we integrated the NS-3 module of Device Language Message Specification - Companion Specification for Energy Metering (DLMS-COSEM), that follows the IEC 62056 standards for electricity metering data exchange. This module which comprises of application and transport layers works in addition with the physical and MAC layer of the

C-UNB module.

Since wireless communication is prone to eavesdropping and information leakages, it is crucial to conduct security studies on these networks. Hence, we performed some cyber-attacks such as Denial of Service (DoS), Address Resolution Protocol (ARP) spoofing and Man-in-the-Middle (MiTM) attacks in the testbed, to analyze their impact on normal operation of AMI network. Encryption techniques can alleviate the issue of data hijacking, but makes the network traffic invisible, which prevents conventional Intrusion Detection Systems (IDS) from undertaking packet-level inspection. Thus, we developed a Bayesian-based IDS for ARP spoof detection to prevent rogue smart meters from modifying genuine data or injecting false data.

The proposed real time simulation testbed is successfully utilized to perform delay and throughput analysis for the existing wireless technologies alongwith the evaluation of the novel features of C-UNB module in NS-3. This module can be used to evaluate a broad range of traffic. Using the testbed we also validated our IDS for ARP spoofing attack. This work can be further utilized by security researchers to study different cyber attacks in the AMI network and propose new attack prevention and detection solution. Moreover, it can also allow wireless communication researchers to improve our C-UNB module for NS-3.

DEDICATION

This thesis is dedicated to my mother and father, whose affection, love, encouragement and prayers, day and night have made me able to focus on my research.

ACKNOWLEDGMENTS

I have been fortunate to have advisors like Dr. Ana Elisa Goulart and Dr. Karen Butler-Purry, who have been instrumental in motivating me throughout my thesis. Their support to analyze new ideas always helped me try things differently in research. I joined their research group to study an existing real-time testbed developed by the researchers of Power System Automation Lab. Being an engineer for four years I was drawn to the research that involved practical implementation.

Dr. Goulart's penchant for addressing problems in a practical way attracted more towards the work she was undertaking in studying a new wireless technology for smart meter communication. She always communicated ideas clearly, precisely and promptly while resolving my doubts and concerns. I extend my gratitude to her for providing important professional advice that has enhanced my graduate experience. I also deeply appreciate her efforts to give me opportunities to present my research works in various conferences. Those experiences were amazing and exposed me to other researchers working in the field of cyber security. Next, I would like to thank all my committee members for taking the time and effort to conduct my defense and providing useful insights to improve my thesis work. I would like to thank my research mates Joseph Hung Ming, Bo Chen, Ogbonnaya Bassey for continuously guiding me.

A graduate experience without great friends is incomplete. I have been able to make great friends, involve in productive discussions and build wonderful memories throughout my graduate school experience. Specifically, I have had my best times with Abhishek Deb, Amarnath Mahadevuni, and Sagar Samant, who have been great roommates for two years.

I deeply thank the university and department committees for having admitted me into the Master of Science (M.S.) program in the Department of Electrical and Computer Engineering and providing a great international exposure. Ultimately, I thank my parents, my family and the Almighty for their blessings and continuous support.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Dr. Karen Butler-Purry of Electrical and Computer Engineering Department, Dr. Ana Goulart of Engineering Technology and Industrial Distribution Department, Dr. Pierce Cantrell and Dr. Le Xie of Electrical and Computer Engineering Department.

Majority of the work conducted for the thesis was done independently by me under the guidance of Dr. Ana Goulart and Dr. Karen Butler-Purry.

Funding Sources

Graduate study was supported by P.I. incentive account of Dr. Ana Goulart of Engineering Technology and Industrial Distribution Department. An industry research project from Cisco Systems Inc. also funded one semester of my graduate studies.

NOMENCLATURE

3GPP	3rd Generation Partnership Project
AMI	Advanced Metering Infrastructure
ARP	Address Resolution Protocol
ASN	Access Service Network
C-IoT	Cellular Internet of Things
COSEM	Companion Specification for Energy Metering
C-UNB	Co-operative Ultra Narrowband
DER	Distributed Energy Resources
DES	Discrete Event Simulation
DLMS	Device Language Message Specification
DMZ	De-Militarized Zone
DoS	Denial-of-Service
eNB	eNodeB or Base station
FAN	Field Area Network
GSM	Global Standard for Mobile Communication
HAN	Home Area Network
HES	Head End System
HTC	Human Type Communication
H2H	Human to Human

IDS	Intrusion Detection System
IoT	Internet of Things
ISM	Industrial, Scientific and Medical Radio Band
LTE	Long Term Evolution
LP-WAN	Low Power - Wide Area Network
LTE-M	LTE Machine-to-machine
MAC	Media Access Control
MDM	Meter Data Management
MiTM	Man in the Middle
NAN	Neighborhood Area Network
NB-LTE	NarrowBand LTE
NS-2	Network Simulator - 2
NS-3	Network Simulator - 3
OFDMA	Orthogonal Frequency Division Multiple Access
PLC	Power Line Communication
PMU	Phasor Measuring Units
QoS	Quality of Service
RACH	Random Access Channel
RAW	Restricted Access Window
RTDS	Real Time Digital Simulator
RTT	Round Trip Time

TCP	Transmission Control Protocol
TDD	Time Division Duplexing
TDMA	Time Division Multiple Access
TIM	Traffic Indication Map
TWT	Target Wake Time
WAN	Wide Area Network
WAMR	WiMAX Automatic Meter Reading
Wi-Fi	Wireless Fidelity (IEEE 802.11)
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

The electric grid is transitioning to a smart grid that employs advanced communication technologies. In this smart grid, Advanced Metering Infrastructure (AMI) network plays an important role in collecting information from smart meters, Phasor Measuring Units (PMUs) and sensors. With advanced computing and communications, cybersecurity has emerged to be a critical issue for AMI networks. Cyber attackers can steal customers' private information, modify or create false data that can financially impact customers, utilities, and the electricity market.

The motivation to develop a testbed for the AMI network lies in the fact that it creates a cyber-physical environment to understand the impact of cybersecurity events. Also, it provides a platform to evaluate the performance of the communication technologies such as coverage, throughput, and latency. Furthermore, it allows researchers to perform studies on defending and mitigating attacks.

Deciding the extent of approximations is one of the important challenges associated with the design and implementation of the testbed. For example, we need to ensure that the distance between two wireless nodes is less than a few kms to consider the propagation delay to be negligible. There are also challenges with the integration of a novel communication protocol into a simulator. The knowledge of all the Open Systems Interconnection (OSI) seven layers is essential to integrate a new protocol for a specific layer to build the full stack in a host. Moreover, to study the strengths and weaknesses of a network simulator to be deployed in a testbed requires us to understand their mechanisms. Additionally, it is complicated to address the issue of scaling the network for more nodes. To perform security study in the testbed, it required us to learn the functioning of applications like BRO, Snort, ettercap, hming, etc. Besides, it is a challenging task to design an Intrusion Detection System (IDS) for a real-time testbed with real time data and no training data.

1.1 Objective

The initial objective of this thesis is to review different communication systems for the AMI network and to evaluate the performance of various network simulators to be deployed in the testbed. The primary objectives of the thesis are to design a real-time simulation testbed for performance and security analysis and to integrate a cellular-IoT based, Cooperative Ultra Narrowband's (C-UNB) MAC and physical layer model to NS-3. Post integration, we merge the application and transport layer of DLMS-COSEM module of NS-3 to the C-UNB module in the smart meter protocol stack as shown in Figure 1.1. To perform security study in the testbed, we developed a simple Bayesian-based IDS for ARP spoof detection.

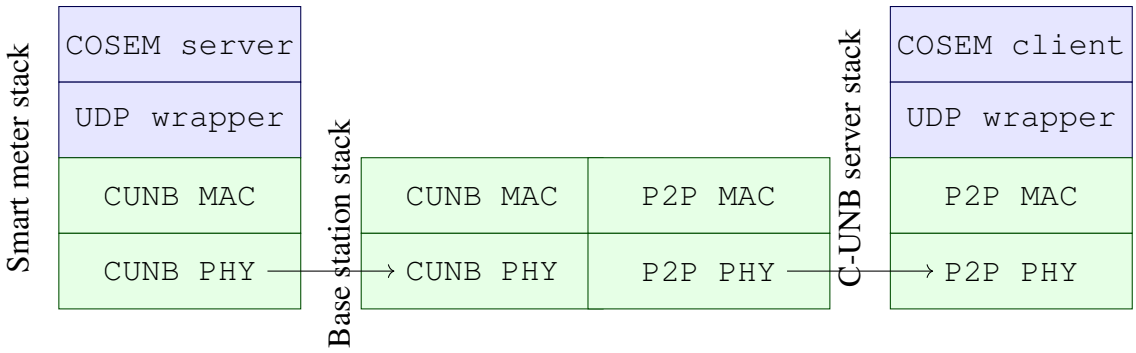

Figure 1.1: Protocol stack of smart meters, base station and C-UNB server.

1.2 Introduction to AMI network

AMI networks provide two-way communications between utilities and smart meters at the customer side. The utility collects real-time consumption and production information from households and disseminates the real-time prices to its customers. Additionally, household appliances can be remotely-controlled by the utility for load management purposes.

According to IEEE 2030 standards [1], an end-to-end smart grid communication model is

shown in Figure 1.2. It includes various communication networks, depending on the geographical range and purpose. The Wide Area Network (WAN) connects data concentrators with the Head-End System (HES) in Core network. The Neighborhood Area Network (NAN) connects smart meters to the data concentrators. Smart meters are connected by long-range and high bandwidth links such as fiber optics, Power Line Communications (PLC), Worldwide Interoperability for Microwave Access (WiMAX), satellite or cellular networks. The Home Area Network (HAN) serves as the communication infrastructure for sensors and devices inside a house. At last, the Field Area Network (FAN) connects feeders, microgrids, and distribution substations to the utility's control and operation center.

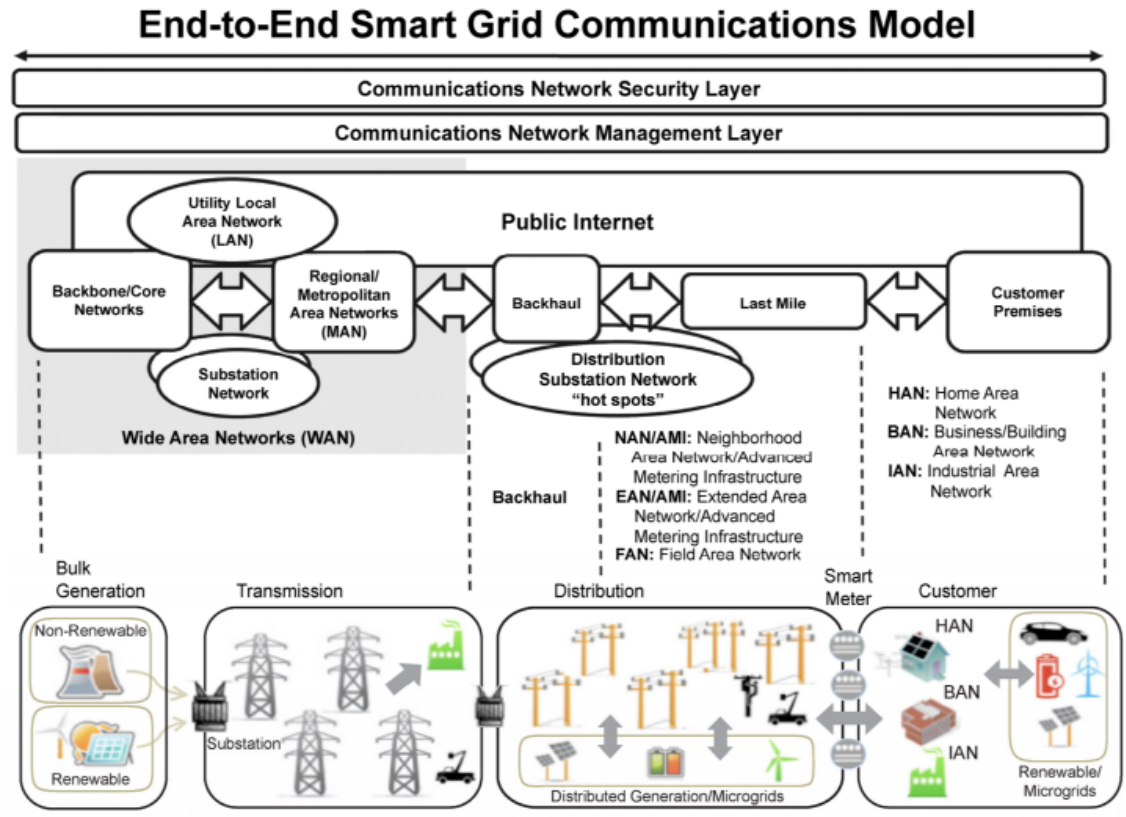

Figure 1.2: End to end smart grid communication model. Reprinted with permission from [1].

1.3 AMI architecture

The AMI architecture is commonly structured into communication networks and consists of head-end system, data aggregator, relays and smart meters, as shown from left to right in Figure 1.3. Each component of the AMI network is explained below.

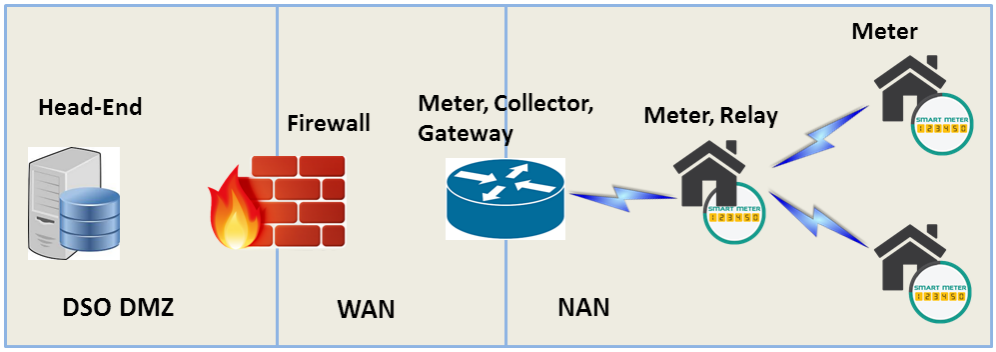

Figure 1.3: AMI components. Adapted with permission from [2].

1.3.1 Head-End System

The HES communicates with the smart meters and is located in a demilitarized zone (DMZ). The collected data from the customer is stored in a Metering Data Management (MDM) system for billing purpose and also forwarded to the Demand Response (DR) system.

1.3.2 Data aggregator or concentrator

The data aggregator acts as the communication node for the HES. It interfaces the HES in the Core network and the smart meters and other aggregators in the NAN.

1.3.3 Smart meters

The smart meters, installed at customers' house, generates electricity consumption data, which when connected with an aggregator can directly communicate to the HES.

1.3.4 Communication

The infrastructure consists of several networks, such as WAN, NAN, and HAN as mentioned earlier. The WAN connects an aggregator, a relay agent or a meter to the HES. Communication on the WAN uses the Internet protocol (IP) network. The primary focus of our thesis is on modeling the NAN of the AMI network.

1.4 Literature review of communication technologies

Different bandwidth and latency requirements diversifies smart grid applications. For instance, AMI traffic which generates small sized payload, requires lower bandwidth with a relaxed latency conditions whereas the automation system of substation requires higher bandwidth and stringent latency [7]. It is not only the availability, the reliability of the network is a vital feature for substation automation and Supervisory Control And Data Acquisition (SCADA). The usage of standardized technologies ensure better scalability and interoperability. The communication technologies must be designed in a way that ensures resiliency to cyber attacks. The preference for a technology depends upon the applications' requirement. Latency, security, and reliability will be the primary criteria for selecting a communication technology for critical applications like SCADA and substation automation. The infrastructural cost does not affect the selection process. But for applications like AMI and electric vehicles that are non-critical, cost will be a deciding factor since they are deployed in a large quantity.

From financial standpoint, wired technologies such as PLC and fiber optics have not been satisfactory for NAN of AMI. PLC failed due to its high latency and low bandwidth [8], and fiber optics due to its high cost for large scale deployment.

Wireless technologies are associated with numerous challenges such as electromagnetic interference. Due to interference from a high voltage electrical equipment communications is impaired [9]. Moreover, interference from other wireless technologies will result in adverse effects like fading on transmissions. However, improvements in acknowledgment techniques, retransmis-

sion policies, error correction and detection algorithms have enhanced the wireless communications' reliability. Hence, with the increasing number of smart meters and new demands for shorter report intervals, there is a need to study existing wireless technologies that can be used in NAN.

1.4.1 IEEE 802.16 (WiMAX)

WiMAX is a wireless broadband technology providing long distance services up to 10 km with a data rate up to 100 Mbps [10]. It qualifies as one of the preferred solution for Wireless Automatic Meter Reading (WAMR), as part of a utilitys' AMI network because of its coverage and high data rates. This standard supports quality of service (QoS) and real-time two-way communications between nodes [10]. A WiMAX system comprises of WiMAX base station and WiMAX receiver in the smart meter. It has a Time-Division Duplexing (TDD) frame structure, where a frame is divided into uplink and downlink sub-frame. The base station sends media access downstream messages related to modulation and coding based on channel conditions [11], at the beginning of the frame. In the upstream, smart meters send the information regarding the time duration they will occupy the channel. Based on the traffic information, the ratio of uplink/downlink slots is configured by the base station. For example, the base station allocates more time slots to uplink when smart meters are sending data, and more slots to downlink when smart meters are downloading upgrades [6]. According to an example in [11], each smart meter reporting measurements once within a time slot, a WiMAX channel can theoretically support 24,000 to 48,000 metering devices. Yet, there are some impediments associated with WiMAX. For example, WiMAX towers comprise of costly radio equipment, resulting in high infrastructure expenses. Moreover the lease of the licensed lower frequency bands from third parties is an overhead.

1.4.2 IEEE 802.11 (Wi-Fi)

The IEEE 802.11 standard refers to a list of wireless technology widely known as Wi-Fi used for Wireless Local Area Network (WLAN) networks. There are a series of standards for IEEE 802.11, such as 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac. It usually functions in 2.4 GHz and

5 GHz unlicensed Industrial, Scientific and Medical (ISM) frequency bands. The data rates of IEEE 802.11x standard ranges from 2 Mbps to 600 Mbps, and the coverage ranges up to 100 m [10]. But in the NAN of AMI, the primary concern is the coverage and the maximum transmit power regulations of the smart meters and not the latency. Hence, we considered IEEE 802.11ah which combines the advantages of conventional Wi-Fi and low-power communication technologies like Sigfox [12]. The IEEE 802.11ah utilizes the unlicensed sub-1 GHz frequency bands as the carrier, for example 863 to 868 MHz in Europe and 902 to 928 MHz in North America [13]. It facilitates communication up to 1 km with a data rate of 150 kbps [14] and a higher coverage than existing Wireless Personal Area Network (WPAN) and better throughput than Low Power Wireless Area Network (LP-WAN) technologies like Sigfox and LoRA. It provides support for densely populated energy-constrained meters by introducing functionalities in its MAC layer, such as fast association, short MAC header, hierarchical organization, Restricted Access Window (RAW), Target Wake Time (TWT), and Traffic Indication Map (TIM) segmentation [12].

1.4.3 Long Term Evolution (LTE)

In earlier days, utility providers considered LTE for AMI backhaul network, but a number of obstacles needed to be lifted before LTE-based smart meters and data concentrators could be deployed. The cost and spectrum are the two limiting factors that prevented utility companies from adopting private LTE networks. The connection-oriented approach in the traditional cellular networks generate enormous signaling overhead [15] for transmitting smaller payloads by IoT devices such as smart meters. This reduces the net throughput of the channel, as the data associated with AMI traffic is less in comparison to Human-type Communication (HTC).

1.4.4 Cellular IoT

Historically, cellular networks were used for the purpose of Human-to-Human (H2H) communications. Due to the energy constraint, coverage and capacity requirements of the IoT devices, this technology required modification. Hence, the third generation partnership project (3GPP) LTE

released standards such as LTE for MTC (LTE-M) and Narrowband LTE (NB-LTE) [15]. The current LTE-M standard uses a four-step handshaking to resolve all possible collisions among nodes that request channel access as shown in the Figure 1.4. The NB-LTE standard, which operates with a 200 kHz channel bandwidth, has reduced hardware complexity by at least 80 percent in comparison to Category 1 User Equipment (UE) [15].

These Cellular Internet of Things (C-IoT) technologies offer new possibilities to connect thousands of IoT devices such as sensors and smart meters, yet they come with signaling overheads. To address these challenges, 3GPP proposes new physical and data link layer protocols for C-IoT called clean-slate radio access technologies. Among the different clean-slate radio access technologies being defined in the 3GPP technical report, the Cooperative Ultra Narrow Band (C-UNB) proposes a novel way to support "massive asynchronous access of small packet transmissions." This is a term defined by the authors in [16], where they have explained the need of a simplified access control scheme for the next generation of smart meters. One of the main objectives of this thesis is to model the MAC and physical layer of the proposed C-UNB technique in NS-3. Section 2 and Section 4 of the thesis explains the technology and its design in NS-3 respectively.

1.4.5 PLC

PLC is a communication technology that utilizes power cables to send data. It has gained importance as AMI backhaul network since it doesn't require extra cabling. Some researchers like Kerk [17] and Soh [18] justified that PLC AMI used along with wireless technologies is the best solution for less tariff price in India and Singapore. The smart meters uses the RS232 data port to connect with a PLC modem [11]. A collection of these modems within the same pole transformer, connect to a single concentrator modem. This concentrator bridges the PLC network with the data network. Power lines are designed with the objective of delivering electricity, hence the complex distribution network and noisy environments may cause interference to PLC, distorting the communication signals. For example, PLC signal voltages coupled to the power lines are negatively

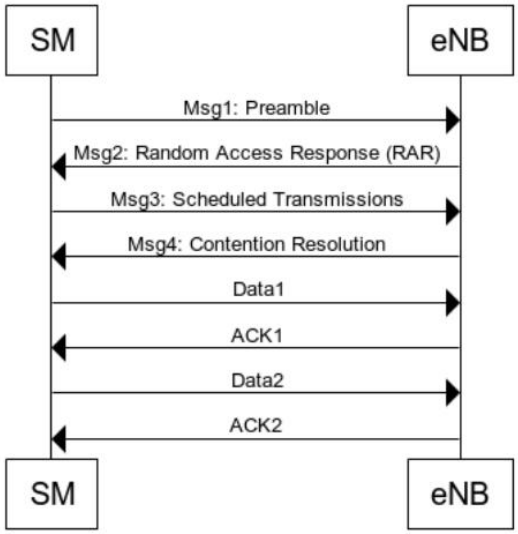

Figure 1.4: The four-way handshake of LTE-M. Adapted with permission from [3].

affected during the change in the impedance of load. Similarly, the switching of electrical devices on a power distribution network may cause power-related parameters, culminating in PLC signal attenuation. PLC's Orthogonal Frequency Division Multiplexing (OFDM) technology, addresses the selective carrier frequency attenuation issue by employing multiple carrier frequencies. However, its higher peak-to-average power ratio causes additional problems, resulting in signal power being averaged down in comparison to a single carrier frequency [19].

1.5 Literature review of network simulators

Network simulators are predominantly used for the evaluation of novel communication architectures and protocols. It allows one to design networks by specifying the behavior of the nodes and the communication channels. For instance, to evaluate the performance of a new scheduling algorithm for MAC layer, it is designed and tested using a simulator. Most of the existing simulation toolkits come with the functionality of Discrete Event-based Simulation (DES). Yet some of them have the shortcomings with respect to scalability, performance and ease-of-use. This leaves many researchers with the conundrum of selecting an apt network simulator for their test-cases.

Therefore, it is necessary to review different simulators before using it in the testbed.

1.5.1 Network Simulator-2 (NS-2)

NS-2 is a DES that provides a platform for the simulation of User Datagram Protocol (UDP), Transmission Control Protocol (TCP), routing, and MAC scheduling protocols over wired and wireless technologies. It is predominantly used by researchers. But its design trades off simulation performance for the saving of recompilations, which is a concern if one is focussed towards performing scalable network simulations [20].

1.5.2 Network Simulator-3 (NS-3)

The integration of real implementations code is supported in NS-3 by providing standard Application Process Interfaces (API), such as Berkeley sockets or Portable Operating System Interface (POSIX) threads [21], which are easily mapped to the simulation environment using NS-3 Terminal Access Point (TAP) and Emulator (EMU) models. TAP APIs are used to connect end-to-end physical systems through the simulated network model, while EMU is used to model simulated nodes with real time network. Additionally, NS-3 integrates architectural concepts and code from Georgia Tech Network Simulator (GTNetS) [22], which enhances its scalability and makes it a better solution than its predecessor NS-2.

1.5.3 OPNET

OPNET Modeler is a well-known commercial DES, with an advanced graphical user interface utilized for the creation of models, data analysis, and the execution of simulation. But the external modules cannot be developed since the kernel is not open source as stated by [23] and it is complicated to design a specific component in the simulator [24]. Although it provides a simulation-in-the-loop feature to inject real time traffic into the simulated network, it requires licensing to use LTE and other wireless modules.

1.5.4 OMNET++

Unlike NS-3 and NS-2, OMNeT++ can only be considered as a general purpose DES framework and not a network simulator . Mostly it is utilized in network simulation due to its INET package that issues a list of IP models. But as stated in [20] it consumes a lot of memory in comparison to NS-3 and NS-2.

1.5.5 SimPy packet generator

Unlike other network simulator, SimPy do not facilitate different widely known network models like Wi-Fi, WiMAX, or LTE. It is a python-based bare simulation API. The processes are the basic simulation components in SimPy, which provide a platform to run multiple processes in an infinite loop. So, we executed the processes of sending data from smart meters and acknowledgments (ACKs) from base stations as two processes, one triggering the other.

1.6 Literature review of protocols for smart metering

The selection of a protocol for smart meter communication is based on some relevant criteria such as openness of the standard, scalability, position in OSI layer and functionality [25].

1.6.1 DLMS-COSEM or IEC 62056

The objective of a communication protocol is for inter-operability among energy devices to exchange information using diverse physical media. One of the metering standard, that support communication for gas, electricity, and water meters is DLMS/COSEM [26]. The DLMS is an application layer protocol, that defines the concept related to the modeling of object-oriented services, communication protocols, and entities [25]. The COSEM represents an object oriented model for the smart meters. In COSEM, the physical meter is considered as a set of logical devices each having a unique identifier and holding information that are designed by interface objects [11]. The methods and attributes of these objects are accessed using a extended DLMS (xDLMS) application layer protocol. An attribute describes the aspects of the data, and methods are utilized to read or

modify them. It provides a standard code, Object Identification System (OBIS) code, to reference all the information associated with a smart meter [27]. These codes are hierarchically organized using six groups based on energy types, physical data items and historical values. The exchange of data between the smart meter and the data collection system follows a server/client paradigm with the collection system acting as client and the smart meter as server [11]. For example, the client application would send the OBIS code such as "1.0.1.8.0.255" and the server would respond "489.8 kWh".

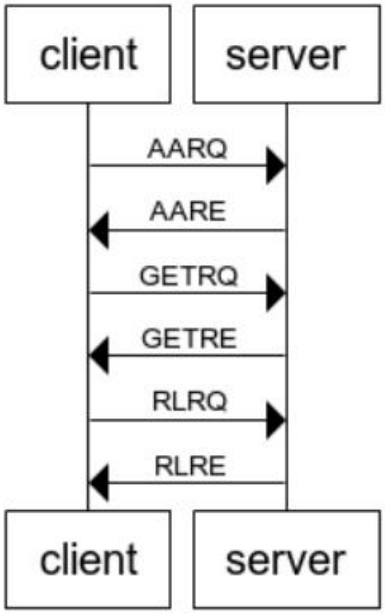

Figure 1.5: Application association and release between DLMS-COSEM server and client.

The DLMS-COSEM module of NS-3 does not have the physical and High-level Data Link Control (HDLC) layer module [28]. Hence, in our simulation, we considered only the application and transport layers of this protocol. For the application data exchange, an application level connection, called Application Association (AA), needs to be established between a client and a server Application Process (AP), which is handled by the connection-oriented Association Con-

trol Service Element (ACSE). Figure 1.5 shows how a client first associates with the server sending Application Association Request (AARQ) packet. After getting Application Association Response (AARE) from the server it sends GET Request (GETRQ) using the OBIS code. Once the client receives the GET Response (GETRE) it can either release the association by sending Release Request (RLRQ) or can send more GETRQ.

1.6.2 IEC-61850

This standard was primarily developed with intra-substation communication in focus [27], but it can be additionally utilized for metering purposes and for communication between substations or control centers. It is an application layer protocol. Like DLMS-COSEM, it has an object-oriented information model with devices, nodes and classes that store different attributes and functionalities [25].

1.6.3 Smart Message Language (SML)

Unlike IEC 62056 and IEC 61850, the SML defines messages instead of creating an interface object model and services to access it. There is no built-in security features except for the usage of an username and a password field within the SML messages. These messages are transmitted using TCP/UDP over IP networks [25]. So far, SML is confined to Germany [27].

1.6.4 ANSI C12

The American National Standards Institute (ANSI) primarily focuses on developing specifications for AMI standards. The current version is ANSI C12.22 [29], which defines the specification for an application layer and allows transport of AMI application layer messages over any underlying layers like Wi-Fi, PLC or cellular technologies. This protocol is used to transport metering specific data structures, ANSI C12.19 tables. It is predominantly used in USA.

1.7 Literature review of security solutions against ARP cache poisoning

Address Resolution Protocol (ARP) is a basic protocol existing in the data-link layer used to get the MAC address of a host, as per it's IP address. The ARP cache poisoning attack is easily executed using tools like ArpPoison, Ettercap, Cain and Abel, Dniff, ARP-SK. There are different kinds of IDS developed by researchers to detect such attacks. This section provides a literature review of different IDS or modified ARP developed till now.

1.7.1 Stateful ARP

The Stateful ARP proposes the alteration of a default stateless ARP cache to stateful cache [30]. In this scheme, a host maintains queues for the ARP requests and responses. Entries in the response queue are verified with the corresponding one in the requested queue. In this approach gratuitous request/reply is not supported. Besides, this change increases the complexity of the ARP protocol.

1.7.2 Cryptographic solutions

Numerous cryptographic solutions were developed to prevent ARP attacks such as S-ARP and TARP [31]. But these cryptographic features required key management, signature generation and verification which act as an overhead. Moreover, other hosts in the network need to upgrade its network stack to communicate using the new cryptographic based protocol.

1.7.3 Secure unicast ARP

In this approach, the ARP request and reply messages are handled by a Dynamic Host Configuration Protocol (DHCP). The ARP requests are forwarded to a DHCP server [32] and the replies are received from them. This DHCP-based ARP is not viable where static IP addresses are assigned to the nodes.

1.7.4 Bayes-based ARP detection for cloud center

The authors in [33] propose a Bayes-based machine learning technique to calculate the trustworthiness of a host. In the thesis, we proposed a similar algorithm but with different feature

vectors. Their approach was Software Defined Network (SDN) based, to detect attackers in a cloud centers, hence they used a different set of features.

There are many other IDS systems such as BRO and Snort which are well-known in industry and research, but they are mainly protocol specific parsers.

1.8 Outline of the thesis

In this section, we described different components of an AMI network. We reviewed different network simulators, communication technologies, smart metering protocols and IDS solutions for ARP spoof detection.

The remainder of the thesis is organized as follows. Section 2 introduces a new Cellular-IoT based solution for the NAN of the AMI network. Section 3 discusses the creation of the real-time simulation environment in NS-3. The NS-3 design documentation of the novel C-UNB technology introduced in Section 2 is presented in Section 4. In Section 5 we describe the Bayesian based IDS proposed for ARP spoof detection. Section 6 comprises of the results and analysis of real-time simulation testbed, the proposed C-UNB module and the IDS solution. Section 7 concludes the thesis and discusses potential future works that can be extended from our current work.

2. CELLULAR IOT BASED, C-UNB FOR THE AMI NETWORK[1]

2.1 Introduction

Remote monitoring is employed in various fields such as energy, environmental, logistics, and surveillance, where the measurements are transmitted over IP networks. A variety of short-range wireless links and/or public mobile networks can be used to connect sensors to data collection servers [34]. Devices such as smart meters are a part of the distribution/customer domain. They are connected to the smart grid either by direct connections to the data collection center, or through one or more concentrators or relay agents. In the later approach, each neighborhood may have one or more concentrators, as shown in Figure 2.1. Smart meters can be connected to the concentrator using proprietary wireless mesh technologies or access protocols such as IEEE 802.15.4g and 6LoWPAN. It forms a Neighborhood Area Network (NAN). In the research by Berthier [35], a NAN consisting of 30,000 smart meters and 90 concentrators was studied using real-world packet traces. While the smart meters-to-concentrator links used proprietary communication protocols, the concentrator relays the information to the data collection server using public mobile networks (i.e., cellular links).

On the other hand, the direct connections approach uses public mobile networks to connect smart meters directly to the data collection server, as shown in Figure 2.2. An overview of machine-type-communications in public mobile networks is described in [36]. It lists the main requirements to trigger a connection, for instance, from a collection service located in another network trying to reach a device inside the mobile network. Essential requirements are the identity of the device and the application, a sequence number that can be used to detect duplicate packets and send acknowledgments, and other optional requirements. In particular, Cellular IoT (C-IoT) offers new

[1]Parts of the material presented in this section are reprinted with permission from "Cellular IoT for Mobile Autonomous Reporting in the Smart Grid" by A. Goulart and A. Sahu. *International Journal of Interdisciplinary Telecommunications and Networking*, July 2016. ©[2016] by IJITN.

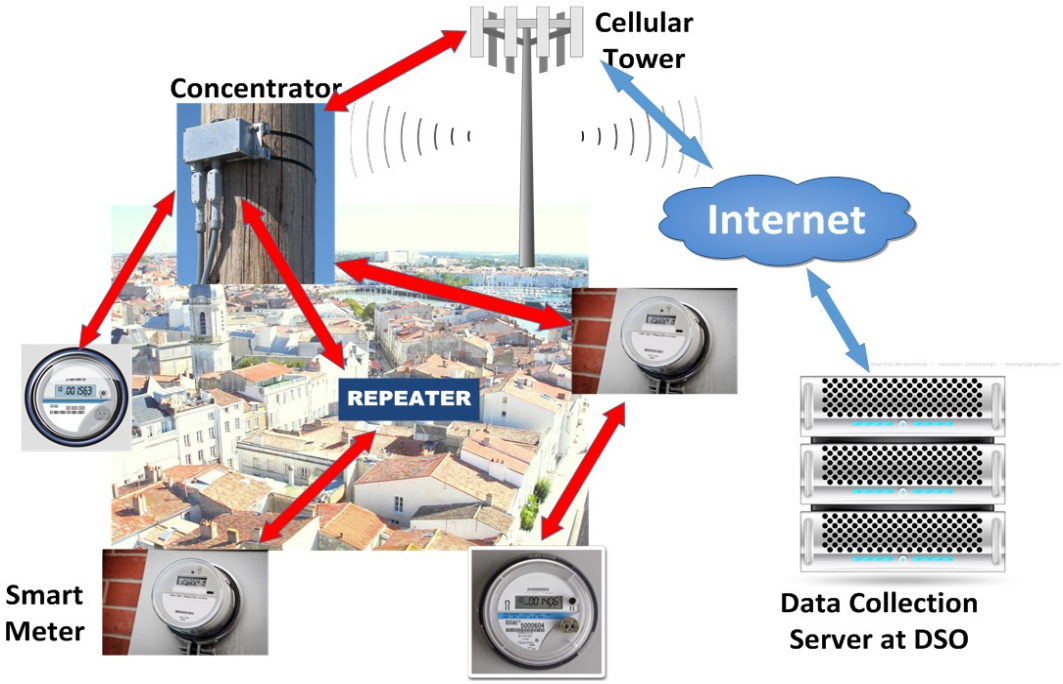

Figure 2.1: Indirect approach of Neighborhood Area Network. Reprinted with permission from [4].

possibilities to connect thousands of IoT devices such as sensors and smart meters using cellular networks.

The goals of the clean-slate radio access technologies, mentioned in the Section 1.4.4 are to support low-bandwidth, low-cost devices, long battery life, and extended coverage. For clean-slate radio access, the standards are being developed to support different modes of transmission such as:

1) Narrow band M2M (N-M2M)

2) Narrow band OFDMA (N-OFDMA)

3) Narrow band cellular IoT (N-CIoT)

4) Cooperative ultra narrow band (C-UNB)

Table 2.1 provides a high-level comparison of the clean-slate technologies that are being developed [4]. All of them use a 200 kHz channels, with 10 kHz guard bands on each side with effective

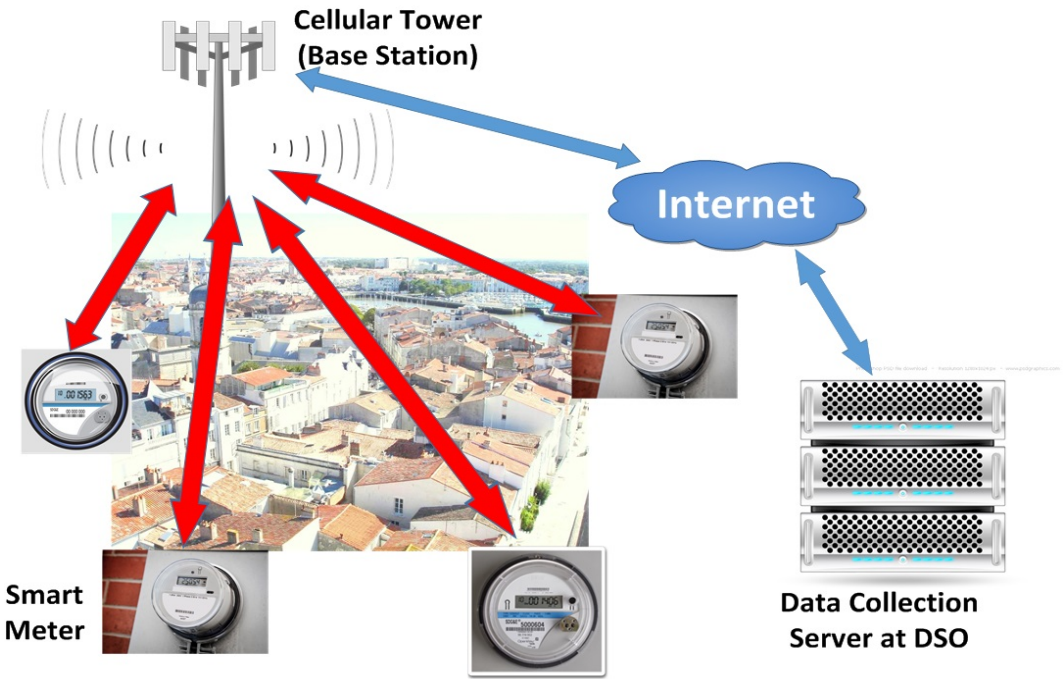

Figure 2.2: Direct approach of Neighborhood Area Network. Reprinted with permission from [4].

bandwidth of 180 kHz [4]. Among these technologies C-UNB was considered for the NAN of the AMI network.

2.2 C-UNB

In C-UNB, the extended coverage is achieved by the use of ultra narrowband channels, which are possible because of the low volume of data transmitted by mobile stations. The C-UNB Radio Access Technology (RAT) is based on random transmissions by mobile devices, which reduces the complexity of C-UNB devices. Access to the uplink channel is a grant free protocol like ALOHA, where C-UNB devices are not required to be assigned resources before transmitting their radio packets. The major drawback associated with such transmission is higher collision rates which are addressed by two mitigation techniques that are frequency and space diversity.

Table 2.1: Overview of 3GPP clean slate radio technologies for cellular IoT. Reprinted with permission from [4].

Technology	System BW	Multiplexing Scheme	Uplink	Downlink	Access Control
N-M2M	180 KHz	FDMA/ TDMA	36 channels	12 channels	RACH carries MS request to send uplink data.
N-OFDMA	180 KHz	OFDMA/ TDMA	72 sub carriers	72 sub carriers	RACH carries MS request to send uplink data.
N-CIoT	180 KHz	OFDMA for downlink	36 channels	48 sub carriers	RACH carries MS request to send uplink data.
C-UNB	180 KHz	FDMA	> 100 channels	> 100 channels	MS selects uplink data channel randomly.

2.2.1 C-UNB system architecture

The C-UNB system architecture comprises a collection of mobile stations, base stations (BS), and a dedicated C-UNB server. The base station primarily implements the C-UNB air interface and acts as the gateway between the C-UNB air interface and the C-UNB server in the core network. The server is unique in a specific C-UNB network. In the uplink, it removes duplicate packets, manages the link layer, and performs security check. For the downlink, it stores the message until a smart meter wakes up. It also performs other functionalities such as segmentation, authentication and base station selection.

2.2.2 C-UNB physical layer

C-UNB channels are created by dividing 200 kHz channels into channels of very narrow bandwidth. Although the specific bandwidth is not defined in [5], given that the uplink bit rate is estimated to be 250 bps and the modulation scheme is Differential Binary Phase Shift Keying (D-BPSK), we assume that the required bandwidth of each micro-channel is 500 Hz, i.e., two times the bit rate. This is equivalent to 360 uplink micro-channels. The C-UNB downlink also has ad-hoc

micro-channels. The downlink bit rate is 600 bps. The packet sizes for both downlink and uplink can have a payload of variable length, so the transmission time of a base station varies. There are also periodical beacon transmissions sent by the base stations (at a higher bit rate of 8 kbps) in case they have to send some information and the sensors have not transmitted for some time.

2.2.3 C-UNB access control layer

The traditional access control of 3GPP LTE systems is based on the idea that the network has a number of designated channels for uplink transmissions, and when a device enters the network or needs to allocate a new channel for transmission, it needs to request access using a Physical Random Access Channel (PRACH). For Machine-to-Machine (M2M) communications in LTE networks, there are many devices that may transmit simultaneously and this may result in congestion due to insufficient capacity. The work in [37] evaluated the random access procedure of the downlink control channel in LTE networks for M2M communications using a parallel Slotted ALOHA model and they showed that the LTE signaling is a bottleneck for a large number of devices.

In the C-UNB access control scheme, the device does not need to request access using PRACH. It selects a random uplink channel and transmits directly. The C-UNB server, which acts as a concentrator or relay, receives data from the IoT devices, and stores data that should be sent to them. Because of its simple access control mechanism and the introduction of the C-UNB server, the work in this thesis investigates the use of C-UNB to provide direct connections between IoT devices, such as smart meters (SM) and sensors, with the data collection server in a Distribution System Operator (DSO).

In C-UNB, the IoT devices can transmit at once to the base station, without the need of any access control protocol. It is a contention-based data link protocol, similar to unslotted ALOHA, but with hundreds of channels. When an uplink packet is received, a C-UNB server sends an acknowledgment using the same micro-channel that was used in the uplink. If a collision is detected,

i.e., two devices transmit in the same micro-channel and no acknowledgment is received, then the frame is retransmitted after a random back-off. In other words, retransmissions at the link layer are supported. The uplink header has a repetition counter. It can only have a maximum three retransmissions to reduce collision rates. For Mobile Autonomous Reporting (MAR) applications, it is assumed at most two repetitions, and three repetitions are said to be normal for exception reports (e.g., alarms). But its drawback is higher packet collision, which is reduced to increase its efficiency by using frequency diversity and space diversity techniques.

2.2.4 Frequency diversity

In a 200 kHz block spectrum, ultra narrowband transmission occupies a very small portion, leaving space for other transmissions. In the case of C-UNB, there is no channelization of the 200 kHz block, i.e., the center frequency of each transmission is selected randomly by a transmitting mobile station in a 200 kHz block. Out of the 200 kHz, the useful bandwidth is 180 kHz, with two 10 kHz guard bands. The *frequency diversity* characteristic is this random selection of carrier frequency that helps to reduce the radio packet collisions. This uplink implementation is called "ad-hoc micro-channel" in C-UNB solution.

2.2.5 Space diversity

There are multiple receptions of radio packets in the uplink. All base stations listen to the same 200 kHz block continuously. Based on the location of the base stations, several of them may receive the same radio packet. If a base station is unable to receive a packet either due to its low received power or due to the collision of the uplink packet, another base station will be able to process the data. In cases where more than one base station successfully receive the packet, deduplication is performed on the C-UNB server. The C-UNB server then acknowledges the packet through the base station that received the uplink packet with higher reception power and is not busy transmitting or receiving other packet. This characteristic of the C-UNB network to cope with collision is called the *space diversity*.

2.2.6 C-UNB server

The C-UNB server is an intermediate or liaison node between the radio access network and the cellular providers' core network as shown in Figure 2.3. A C-UNB server receives and stores data or requests from the collection server, and sends it when the device wakes up (from idle or power-saving mode). It listens for the devices' uplink transmissions, acknowledges it and sends any stored data at the same time. The C-UNB server also checks the authentication information sent by the devices, which is in the form of a 16-bit hash value (Figure 2.4). The hash is computed using the devices' identity information with the frames' sequence number and payload bits, using a secret key that is shared by the device and the C-UNB server. The same hash value is used in the downlink acknowledgment frame.

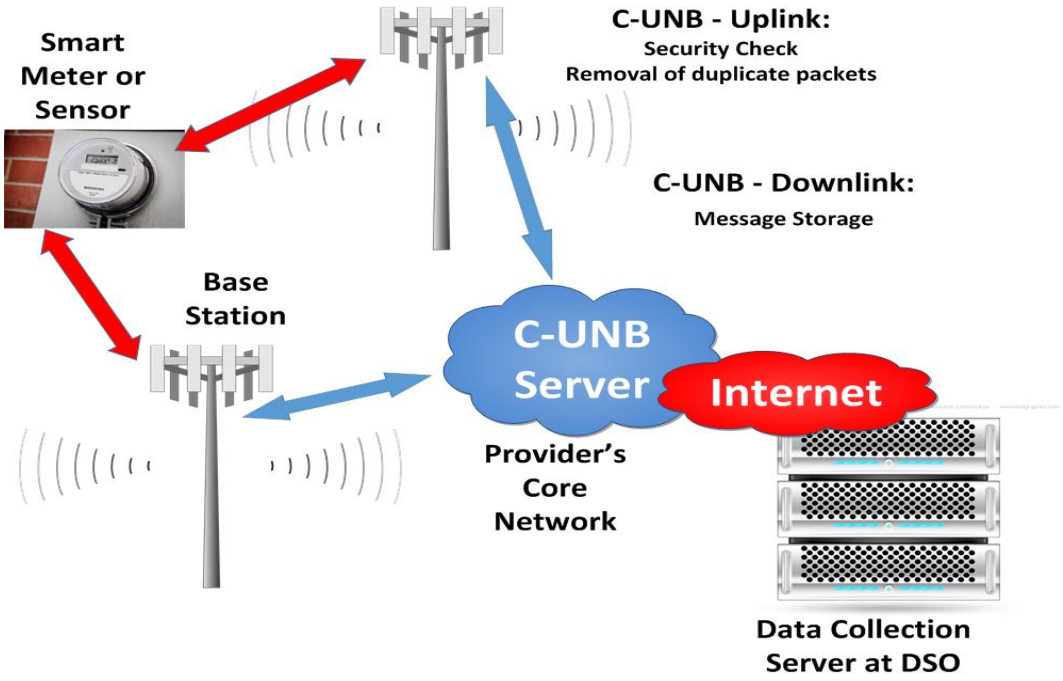

Figure 2.3: C-UNB server in the DSO. Reprinted with permission from [4].

2.2.7 Beacon channels

In C-UNB it is assumed that the 200 kHz block is the same for all base stations of a given mobile network in a given area. They should also transmit in the same frequency block. Each sector of a base station transmits a beacon channel which helps IoT devices to discover the 200 kHz block allocated to C-UNB and send other system information elements. These beacon packets are transmitted periodically. This prevents collision among them. Our simulation considers a single area there will not be any beacon channel packets.

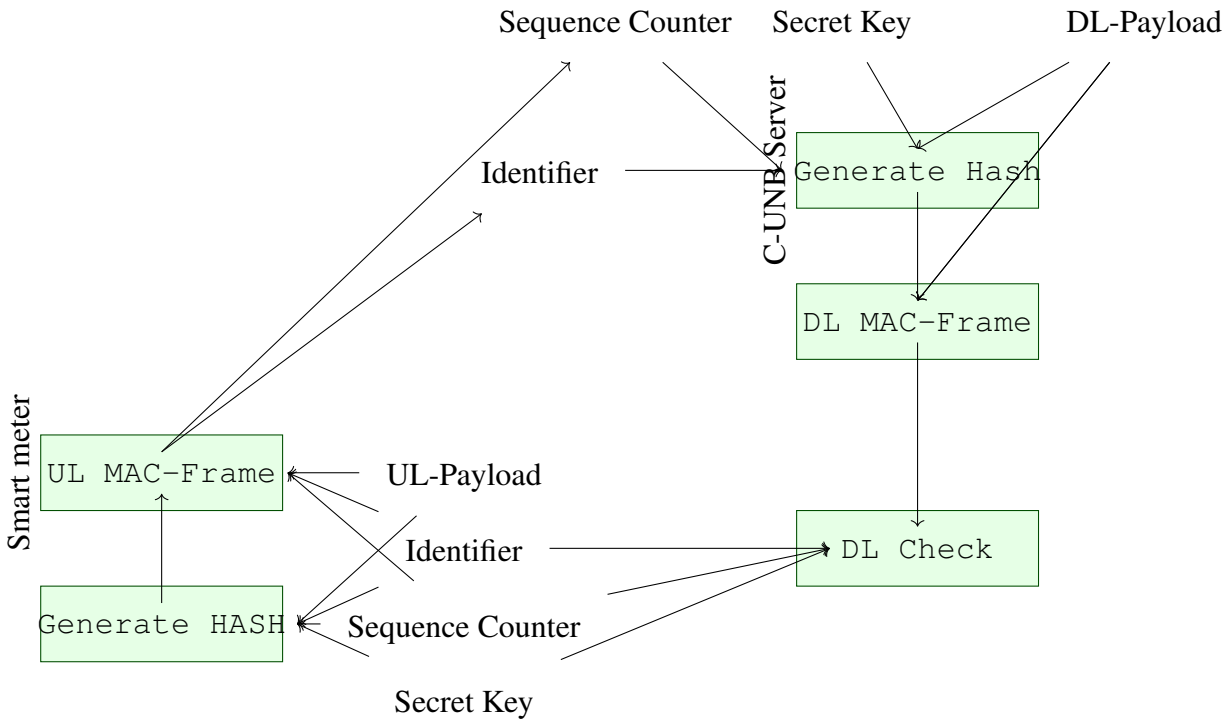

Figure 2.4: Authentication scheme for both uplink and downlink. Adapted with permission from [5].

2.2.8 Uplink packets

The maximum payload size that can be transmitted in a single uplink packet is 32 bytes, but the data size will be much larger than that. Hence, segmentation and reassembly is supported, with a maximum of 31 segments. This segment number and the total number of segments information is present in the header.

Header: The header field is 40 bits long and comprises of a preamble for frame detection and bit rate synchronization, frame type, frame length, acknowledgement flags and repetition counter.

Sequence Counter: The sequence counter field is 12 bits long, which is incremented by one for each new packet sent by device.

Identifier: The identifier field is 40 bits long and contains a unique identifier so that the device can be identified in the C-UNB system.

Payload: The payload ranges from 0 to 32 bytes. This length is defined by a 5-bit length field in the header field. Empty payload implies it is an ACK packet.

Authentication: This is a 16-bit hash field. It is used to authenticate the MAC-Protocol Data Units (MAC-PDU) received by the network. The 16 bits are generated by using C-MAC algorithm. It uses a 128-bit secret key. The authentication verification is performed at the C-UNB server and not on the base station to prevent secret key transmission on the back-haul network. Figure 2.4 displays the authentication process in uplink and downlink transmission.

Frame Check Sequence (FCS): The FCS field is 8 bits long and consists of a Cyclic Redundancy Check - 8 (CRC-8). It is used to check MAC-PDU consistency before transmission to the core network.

Error-Correcting Code (ECC): The ECC field is 16 bits long. It uses Reed-Solomon algorithm and is evaluated over the whole MAC-PDU except the preamble.

Figure 2.5 shows the headers and trailers used in the uplink MAC-PDUs.

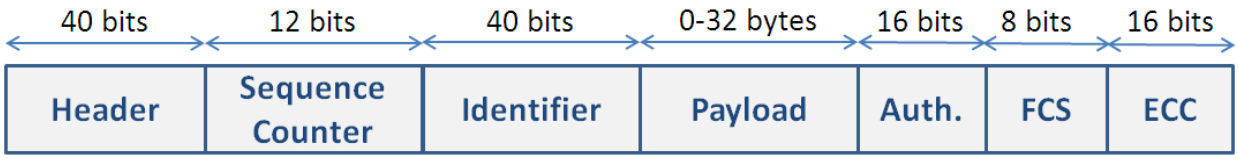

Figure 2.5: Format of MAC-PDU frame in C-UNB uplink transmission. Adapted with permission from [5].

2.2.9 Downlink packets

Downlink MAC-PDUs are transmitted only in response to an uplink transmission. They are also transmitted at the same carrier frequency at which the uplink transmission was received.

Header: The header field is 56 bits long and consists of preamble for frame detection and bit rate synchronization, frame type, payload length, and acknowledgement bits.

Payload: The length of the payload ranges from 0 to 34 bytes. It can carry application data or also the link layer header for segmentation/re-assembly.

Authentication: The authentication field is 16 bits long, which uses C-MAC algorithm for hash calculation. It uses 128 bits secret key of the device along with device identifier, sequence number received in the corresponding uplink packet and the payload field.

FCS: The FCS is 8 bits long and consists of CRC-8. It uses it to check MAC-PDU after error correction before further decoding.

ECC: The ECC is 32 bits long. It uses Bose-Chaudhuri-Hocquenghem (BCH) code applied to header (except preamble), payload, authentication, and FCS fields. Figure 2.6 shows the headers and trailers used in the downlink MAC-PDUs.

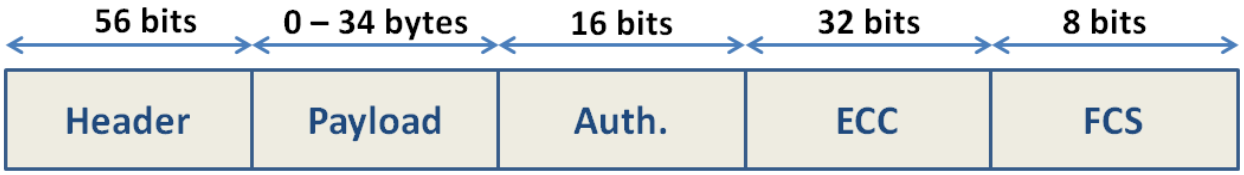

Figure 2.6: Format of MAC-PDU frame in C-UNB downlink transmission. Adapted with permission from [5].

2.3 Retransmission probability calculation

The retransmission of a packet takes place when an IoT device does not receive an ACK in the receive window. There are many reasons that may result in retransmission:

a) Some of the ACKs sent by the C-UNB server may be lost during the downlink transmission.

b) The data packet might be lost during uplink transmission either due to collision with other packet using the same channel or due to lower receiver sensitivity.

In this section, we will analyze the impact of reporting interval and payload size on the retransmission probability.

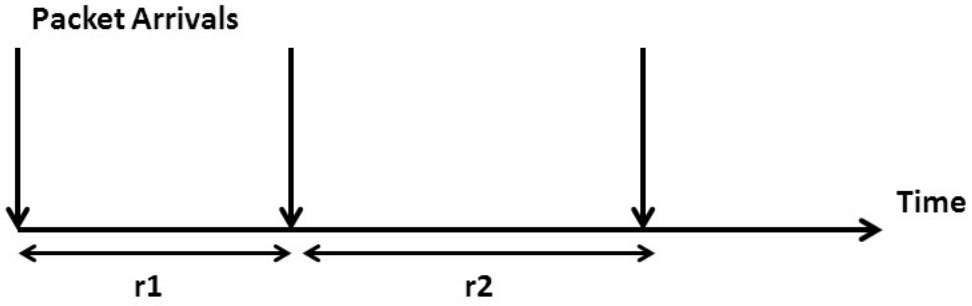

Figure 2.7: Poisson arrival of AMI packets.

Let r1 and r2 be the inter-arrival time between a packet and its immediate predecessor and successor respectively, as shown in the Figure 2.7. The length of the predecessor packet is X1 and

the current packet is X2. Then, Equation 2.1 gives the probability of no collision with any packet.

$$P(\text{no collision with any packet}) = P_1 * P(r2 > X2) \tag{2.1}$$

where P_1 is P(No collision with any preceding packets), which is given by Equation 2.2.

$$P_1 = P(r1 > X1) \tag{2.2}$$

Assuming a fixed length packet, hence the probability of no collision is given by Equation 2.3 [38] and collision by Equation 2.4.

$$P(\text{no collision with any other packet}) = e^{-\lambda(X1+X2)} \tag{2.3}$$

$$P(\text{collision}) = 1 - e^{-\lambda(X1+X2)} \tag{2.4}$$

where, λ is the Poisson arrival rate.

Since the collision happens if both packets use the same frequency, we calculate the probability of both packets using the same frequency, which is given by Equation 2.5.

$$P(\text{both packets use same frequency}) = 1 - ({}^{s}P_n/s^n) \tag{2.5}$$

where, s is the number of micro-channels, n is the number of smart meters and ${}^{s}P_n$ is given by s!/(s-n)!

Hence, the probability of overlapping packets using the same frequency is given by Equation 2.6.

$$P(\text{collision}) = (1 - {}^{s}P_n/s^n) * (1 - e^{-\lambda(X1+X2)}) \tag{2.6}$$

Before any application layer protocol like DLMS-COSEM initiates the establishment of an application association, the connection between the peer physical layers of the client and server

side need to be established. The physical layer association is initiated by the smart meters by sending an `Hello` packet to the base station. The base station then uses the same channel to forward AARQ request of the C-UNB server, also considered as DLMS-COSEM client. Hence, there are six kinds of packets involved in AMI communication. They are `Hello`, AARE, AARQ, GETRQ, GETRE, and ACKs as shown in Figure 2.8. The Poisson arrival rate of Hello, AARE, AARQ, and GETRQ are negligible in comparison to GETRE and ACK, because they are used only for the initial association. Once the smart meters are associated to the C-UNB server, there is transmission of only GETRE and ACK packets.

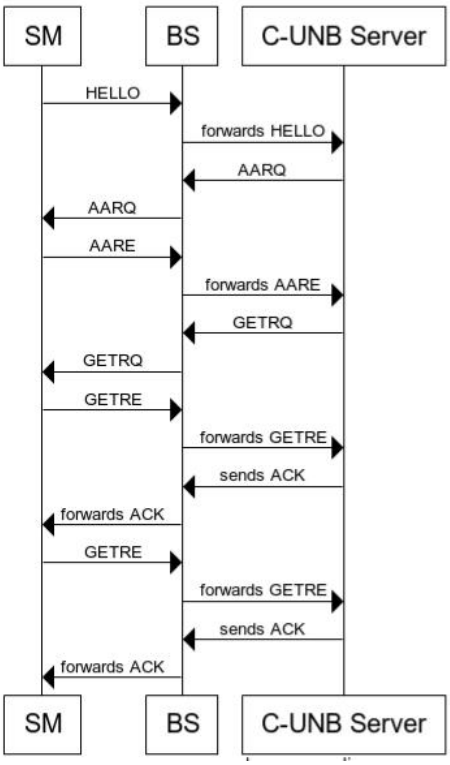

Figure 2.8: Different types of packets.

The probability of a GETRE packet colliding with ACK packet during uplink transmission is

given by Equation 2.7.

$$P(\text{collision between GETRE and ACK}) = (1 - {}^{s}P_n/s^n) * (1 - e^{-(\lambda_{\text{ACK}} T_{\text{ACK}} + \lambda_{\text{GETRE}} T_{\text{GETRE}})}) \quad (2.7)$$

where λ_{ACK} and λ_{GETRE} are the Poisson arrival rates of the ACK and GETRE packets. T_{ACK} and T_{GETRE} are the payload sizes.

Similarly, the probability of an ACK packet colliding with GETRE packet during downlink transmission is given by Equation 2.8.

$$P(\text{collision between ACK and GETRE}) = (1 - {}^{s}P_n/s^n) * (1 - e^{-(\lambda_{\text{GETRE}} T_{\text{GETRE}} + \lambda_{\text{ACK}} T_{\text{ACK}})}) \quad (2.8)$$

Though the events for Equation 2.7 and 2.8 are same but they cause two different retransmission hence from the reference of retransmission they are two mutually exclusive events.

But a GETRE packet can also collide with another GETRE packet whose collision probability is given by Equation 2.9 and an ACK packet can collide with another ACK packet whose collision probability is given by Equation 2.10.

$$P(\text{collision between GETREs}) = (1 - {}^{s}P_n/s^n) * (1 - e^{-2 * \lambda_{\text{GETRE}} T_{\text{GETRE}}}) \quad (2.9)$$

$$P(\text{collision between ACKs}) = (1 - {}^{s}P_n/s^n) * (1 - e^{-2 * \lambda_{\text{ACK}} T_{\text{ACK}}}) \quad (2.10)$$

The overall probability of retransmission is given by Equation 2.11.

$$\begin{aligned} P(\text{retransmission}) = P(&\text{none of the base station received the signal}) + \\ &P(\text{ACK packet is lost}) + \\ &P(\text{GETRE packet collided while uplink TX}) \end{aligned} \quad (2.11)$$

The first term in the right hand side of Equation 2.11 is considered to be negligible because of the space diversity feature of C-UNB. The second term is the sum of Equation 2.8 and 2.10. And the third term is the sum of Equation 2.7 and 2.9.

2.4 Channel throughput calculation

In this section we analyze the throughput, S of the network as a function of the network offered traffic, G. For the calculation of throughput, we consider that smart meters i = 1,2,3,....,N create reports at a reporting interval of τ_i s, which occupies the C-UNB channel for $t_{\text{u,i}}$ s for uplink and $t_{\text{d,i}}$ s for downlink. Hence the network offered traffic as described in [39] is given by Equation 2.12.

$$G = \sum_{i=1}^{N} (t_{\text{u,i}} + t_{\text{d,i}})/\tau_i \tag{2.12}$$

The offered traffic refers to the portion of time the channel is utilized for transmission by the smart meter. G < 1 signifies the channel is underutilized. On the contrary, G > 1 implies that the channel is over utilized. For a fixed G, the throughput, S, is given by Equation 2.13.

$$S = G * P_{\text{succ}} \tag{2.13}$$

where, P_{succ} is the probability of successful transmission.

2.5 Summary

This section introduced us to a new C-IoT based solution that utilizes the space and frequency diversity features to address higher collision rates in the random channel access procedure. It explains various component of the C-UNB network along with the calculation of the retransmission probability and the C-UNB channel throughput. Section 4 explains the design of the `cunb` module in NS-3. This module designs the direct connection approach of C-UNB as explained in the Section 2.1. Later in Section 6 we analyze the results obtained using a Python simulator and the `cunb` module in NS-3.

3. MODELING THE AMI NETWORK USING NS-3[1]

3.1 Introduction

NS-3 is developed with the purpose of integrating into a testbed and other virtual machine environments. A real-time scheduler is developed to integrate with the real network stack. The purpose of the scheduler is to control synchronize the progression of the simulation clock with the hardware clock or the external time base [40]. Due to the external time base, the simulation time is prevented from fluctuation of one simulated time to another. The usage of the realtime simulator is quite simple from a scripting perspective, as the users are simply required to set the attribute `SimulatorImplementationType` to `RealtimeSimulatorImpl`. The details of the implementation of the real-time module of NS-3 can be found in [40]. This real-time implementation code of NS-3 can be mapped to the simulation using Terminal Access Point (TAP) and Emulator (EMU) modules of NS-3. The TAP APIs are used to connect end-to-end physical systems through the simulated network, while EMU APIs are used to model the simulated nodes with the real-time network.

3.2 NS-3 TAP interface

For our AMI testbed, the purpose is to simulate communication channels rather than generate data, therefore we adopted the NS-3 TAP model. In NS-3, a bare `Node` is equivalent to the shell of a computer, to which one may add a `NetDevice` or an application [41]. The TAP `NetDevice` object allows real or virtual hosts that support TUN/TAP devices [42] to participate in the simulation. In the testbed, smart meters and data collection servers see the NS-3 node as a gateway to traverse through the communication channel built inside it.

[1]Parts of the material presented in this section are reprinted with permission from "Modeling AMI network for real-time simulation in NS-3" by A. Sahu, A. Goulart and K. Butler-Purry. *2016 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, October 2016. ©[2016] by IEEE.

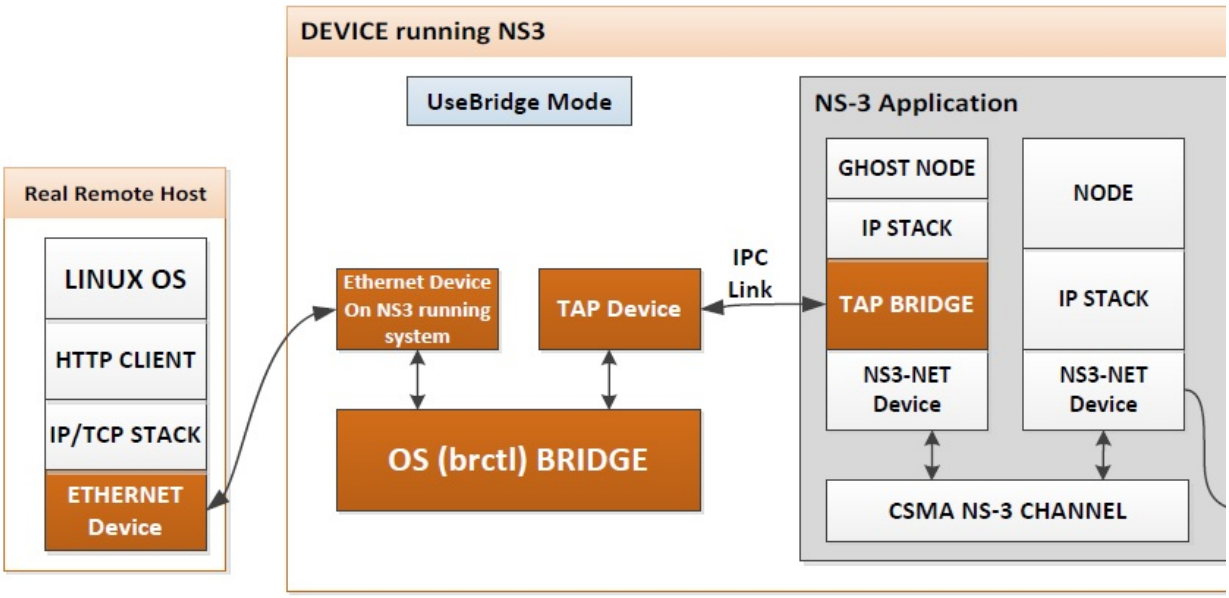

Figure 3.1: NS-3 TAP Bridge `UseBridge` Mode. Reprinted with permission from [6].

The bridging element between the simulated and real TAP devices is called a TAP bridge. The TAP module operates in three modes: `ConfigureLocal`, `UseLocal` and `UseBridge`. In `ConfigureLocal` mode, the creation and configuration like gateway, IP and MAC address of the TAP device is done through simulation. In `UseLocal` mode, the simulation uses TUN/TAP interfaces already configured by the user, but this mode allows only one real `NetDevice` with a unique MAC address to be connected to the Linux bridge. The `UseBridge` mode allows many Linux `NetDevices` on the non-NS-3 side of the bridge to attach. As we model many real devices that send and receive data through a single TAP bridge, we used the `UseBridge` mode in our simulation, which allows many Linux `NetDevices` on the non-NS-3 side of the bridge. Figure 3.1 shows how a physical device is connected to the NS-3 simulator. The Algorithm 1 shows the method followed to create the set up.

Algorithm 1 Configuring TAP interface for real-time simulation.

1: Creation of Linux bridge. Command: `brctl addbr <Bridge Name>`

2: Creation of TAP interface. Command: `tunctl -t <Tap interface name>`

3: Configure the bridge IP address and subnet mask. Command: `ifconfig <Bridge name> <IP address> netmask <subnet mask>`

4: Set the TAP IP to 0.0.0.0 and activate with promiscuous mode. Command: `ifconfig <Tap interface name> 0.0.0.0 promisc up`

5: Add the TAP and real ethernet interface to the bridge. Command: `brctl addif <Bridge name> <Tap interface> <ethernet interface>`

6: Activate all the interfaces. Command: `ifconfig <bridge/tap/ethernet interfaces> up`

7: Do similar set up for client side system.

8: Run the NS-3 simulation script.

9: Start the server and client application in the real devices.

3.3 NS-3 Direct Code Execution (DCE)

NS-3 DCE module acts as an operating system, like loading in memory the code and data of executables [43]. It plays the role of an intermediary between the executable and the simulation through the systems functions triggered by executables. It manages the scheduling of the various virtual processes and threads. It allowed us to execute existing implementations of network protocols or applications inside NS-3 nodes. To understand the mechanism of DCE, we analyzed the `DceManager` class which served as the entry point of DCE. There is an instance of `DceManager` which is associated to each node and virtualizes the exceution of the process.

One can use a HTTP or CoAP (Constrained Application) server and client inside a simulated node through DCE. As LTE devices do not support `UseBridge` mode of TAP bridge, we emulated smart meters with a simulated User Equipment (UE) node in NS-3. DCE was then used to install a HTTP server in this UE node. It communicated with a HTTP client that is running in a physical device outside NS-3.

3.4 Various wireless AMI network model using NS-3

3.4.1 Wi-Fi

Our NS-3 model for Wi-Fi is shown in Figure 3.2, where a point-to-point (P2P) link connects the Wi-Fi network to the core network. The hosts in the core network are considered to be ethernet-based, hence Carrier Sense Multiple Access (CSMA) module of NS-3 is used for their MAC layer. The data collection center hosted in an external machine connects the network using the Linux and TAP bridge described in Section 3.2. Similarly, the smart meter modeled in a Raspberry pi connects the Wi-Fi network. NS-3 includes a `WiFiNetDevice` object following the IEEE 802.11 standard. It supports the basic 802.11 Distributed Coordination Function (DCF) with ad-hoc and infrastructure modes. The Wi-Fi channel model maintains physical state machine, tracks all packets interference, and computes probability of error for a given signal-to-noise ratio (SNR) [44].

At the physical layer, our model has constant propagation delay. Channel fading was modeled using log-normal shadowing as per the Equation 3.1.

$$PL = PL_o + 10\alpha * log_{10}(\frac{d}{d_o}) \tag{3.1}$$

where, α is the path loss exponent, PL is the path loss, PL_o is the path loss at the reference distance d_o and d is the distance between the receiver and the transmitter.

For channel sensing, we used Clear Channel Assessment (CCA) procedure where a medium is assumed busy if the measurement of the sensed power is more than a CCA threshold. The

MAC layer in NS-3, provides several rate control algorithms, such as Auto Rate Fallback (ARF), adaptive-ARF, constant-rate, and minstrel. For simplicity, we used constant-rate, which offers the same transmission mode for both data and control packets. For IEEE 802.11 ac, we adopted a data rate of 6 Mbps and channel width of 160 MHz. For the point-to-point link connecting the Wi-Fi network (10.10.10.0/24) and the CSMA-based core network (11.11.11.0/24), as shown in Figure 3.2, the data rate was set to 1 Mbps and a propagation delay to 8 ms. These parameters can be modified as per the net delay we want to create.

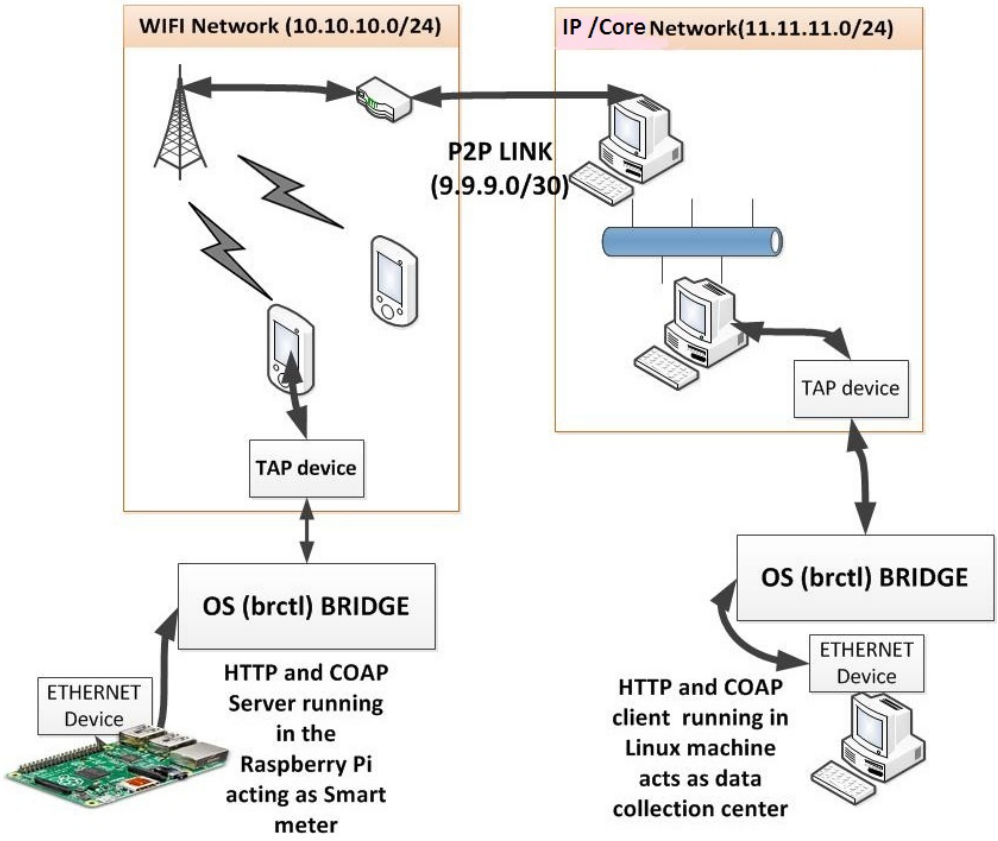

Figure 3.2: Wi-Fi 802.11ah based NAN of AMI for real-time NS-3 simulation. Reprinted with permission from [6].

3.4.1.1 Wi-Fi 802.11ah

The latest version NS-3.25 supports 802.11 ac, and we added an 802.11 ah module developed by University of Antwerp [12]. This newer version of Wi-Fi operates in the 900 MHz band. Recent improvements in this standard [45] have helped to reduce power consumption, extend transmission range, and improve propagation and penetration for M2M communications. Additionally, it does not need to maintain backward compatibility with other 802.11 protocols. The MAC layer of the 802.11 ah provides features such as Restricted Access Window (RAW), Traffic Indication Map (TIM) segmentation and Target Wake Time (TWT), to support a densely populated energy-constrained smart meters [12]. The RAW divides the meters into groups allowing one group to access channel at the same time reducing the probability of collision. Due to TIM segmentation, the information sent in a TIM is splitted into various segments and transmitted separately [12]. The smart meters transmitting data occasionally, TWT causes the reduction of power consumption. TWT-based meters fixes a time slot with the access point to send data based on their state. Due to these supporting features we employed this for our simulation for AMI network. For IEEE 802.11 ah, a constant data rate of 300 kbps and channel width of 1 MHz is configured for NS-3 simulation.

3.4.2 WiMAX

As WiMAX supports non line-of-sight communication (NLOS) with a range of 10 km, it is considered a viable solution to a scattered Neighborhood Area Network (NAN) of AMI. The WiMAX architecture primarily comprises of Access Service Network (ASN) and Connectivity Service Network (CSN). The ASN consists of mainly two parts: base station and WiMAX receiver in the smart meter. The data collection center is considered to be a host in the CSN. A P2P link connects the ASN to the CSN through a ASN Gateway.

NS-3 includes a `WimaxNetDevice` object based on IEEE 802.16 standard. It supports three kinds of uplink and downlink schedulers: a simple First Come First Serve (FCFS), a Real-Time

Polling (RTP) and a Migration-Based uplink Scheduler called MBQoS [46]. The MBQoS scheduler allocates resources based on the Quality of Service (QoS) parameter associated to a service flow and bandwidth requirement of the subscriber station. We used a simple FCFS scheduler in our simulation. A Best-Effort service flow is allocated to both upstream and downstream data. In the physical layer, the channel uses a COST231 propagation and path-loss model. The transmission loss for this model L_b is adopted from the paper [47] given by Equation 3.2

$$\begin{aligned} L_b = 69.55 + 26.16log(\frac{f}{MHz}) - 13.82log(\frac{h_{\text{Base}}}{m}) - a(h_{\text{Mobile}}) \\ +(44.9 - 6.55log(\frac{h_{\text{Base}}}{m})log(\frac{d}{Km})) \end{aligned} \tag{3.2}$$

where, h_{Base} and h_{Mobile} are the height of the base station and the mobile station respectively and d is the distance between them. The function a() in Equation 3.2 is defined by the Equation 3.3

$$a(h_{\text{Mobile}}) = (1.1log(\frac{f}{MHz}) - 0.7)\frac{h_{\text{Mobile}}}{m} - (1.56log(\frac{f}{MHz} - 0.8) \tag{3.3}$$

The modulation and coding scheme (MCS) used in the model is 16-Quadrature Amplitude Modulation (16-QAM) and convolutional coding (n/k ratio = 1:2), with a theoretical data rate of 27.65 Mbps. The attributes like data rate and propagation delay of the point-to-point link between the NAN and the CSN is similar to the one used in the Wi-Fi model.

3.4.3 LTE

Cellular communication provides a fast and reliable connectivity for smart metering infrastructure along with full IP infrastructure, high bandwidth, and low latency. Modern cellular network with ubiquitous reach encourages more development of LTE-M (LTE for M2M). Our NS-3 model for LTE is shown in Figure 3.3. It incorporates two models [48]:

- LTE model: Comprises of the LTE Radio Protocol Stack (Radio Resource Control (RRC), Packet Data Convergence Protocol (PDCP), Radio Link Control (RLC), MAC, physical).

This stack completely resides within smart meters and the base station. Since the LTE model did not support real-time integration like Wi-Fi and WiMAX, we implemented the server in the smart meter using NS-3 DCE.

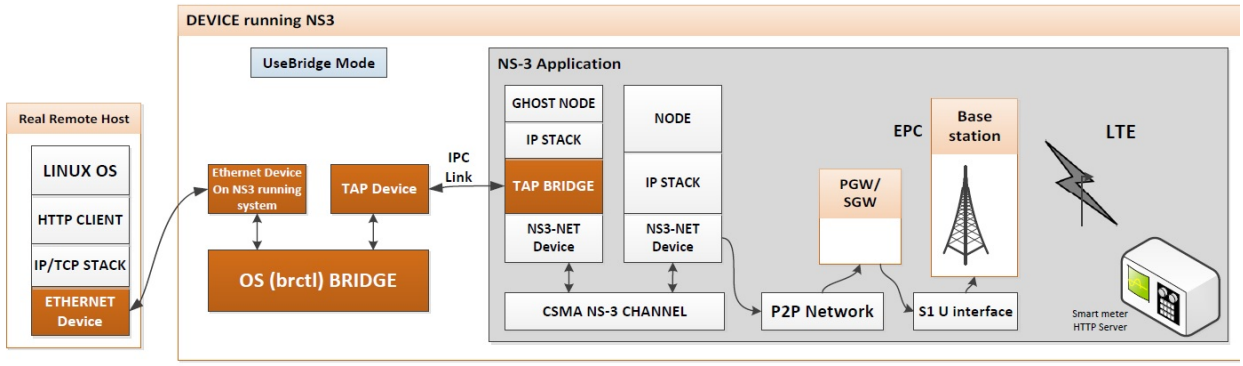

Figure 3.3: LTE model design with remote host as a real computer. Reprinted with permission from [6].

- Evolved Packet Core (EPC) model: Provides the means to simulate end-to-end IP connectivity over the LTE model. The data collection center is connected to the LTE network through the Service Gateway/ Packet Data network Gateway (SGW/PGW) nodes.

The simulator can support up to 10 base stations and 100 smart meters. Currently, we are able to model the LTE-based AMI network with two smart meters and one base station. The detailed explanation of data and control-plane of UEs and eNBs modeled in NS-3 can be found in [48].

3.5 Summary

The real-time NS-3 testbed for end-to-end AMI communication is modeled using the TAP Bridge `UseBridge` mode. The Wi-Fi, WiMAX and LTE modules of NS-3 is used to model the NAN. We will evaluate the performance of these wireless technologies with application layer protocols like HTTP and CoAP in Section 6.3.

4. A NOVEL C-UNB MODULE INTEGRATION TO NS-3[1]

4.1 Introduction

The NS-3 software, an open source DES simulator is used to perform the network simulation. The simulator is extended with the integration of the physical and the MAC layer of a C-UNB module. This section first introduces the NS-3 software and then explains the different classes created for the implementation of this new module. In this section, the word smart meter is used interchangeably with mobile station and the base station with e-NodeB (eNB).

4.2 Network Simulator 3

NS-3 is developed by a community of users, mainly used for research and academic purposes. The object-oriented approach of C++ provides a platform to create different components of a complex network, with each class representing one component of a network. Various classes that model associated concepts are assembled in modules: for instance, the `wifi` module comprises of classes that represent different aspects of a Wi-Fi system, like APs (Access Points), non-AP STAs (Stations), i.e., Wi-Fi enabled smart meters, the Wi-Fi MAC and physical layer and a `YansWifiChannel` object. These classes when merged with the modules representing the core functionality, mobility, propagation etc., creates a full-stack network implementation of the Wi-Fi standard.

NS-3 is considered to be a DES because a single simulation comprises of cascading events, each one scheduled one after another. The simulator executes these events using the callback functions, resulting in a change of the simulation state, followed by scheduling future events. An example of an event is scheduling a series of events, starting with the transmission of a packet to the physical layer from a class modeled as the transmitting devices' MAC layer, which further

[1]Parts of the material presented in this section are reprinted with permission from "Cellular system support for ultra-low complexity and low throughput Internet of Things (CIoT) (Release 13), 3GPP TR 45.820 V13.1.0" by 3rd Generation Partnership Project, November 2015. ©[2015] by 3GPP.

calls the channel, and then schedules an event for reception of the packet by receiving devices' physical layer, after a channel delay, based on the propagation delay model. In some cases, where it is difficult to predict the simulation end time, due to the cascading nature of event creation, a stop event is issued to culminate the simulation. For example, in the `cunb` module we schedule retransmission of the unacknowledged packet and open the receive windows for ACK reception, but when an ACK is received from the C-UNB server, we stop the re-transmission events.

The tracing system of NS-3 is used to check the variable status during the simulation, and if necessary, trigger an action based on the modified variable state. This system collects the data dynamically during the simulation run-time.

4.2.1 Simulation script

The MAC or physical layer models are unique to a network and are developed using different classes in the module. The models and architecture are implemented in a NS-3 simulation, created using an independent script in Python or C++, which follow these steps:

1. Architecture creation: A collection of nodes such as smart meters, concentrators or C-UNB server used in the C-UNB network is created as a `Node` object. These nodes are associated to a `MobilityModel`, that represents the nodes' location and how it alters with time. For example, we used the `ConstantPosition` model for the smart meters and the base stations, since these components are stationary.

2. Network Models: A specific protocol layer object is installed on the set of nodes created in the first step. This is implemented using the helper classes designed for the installation in a node, a particular layers' object in the OSI stack. In this way, a node is able to deal the packets based on that protocol. For example, we created the `CunbHelper`, `CunbMacHelper` and `CunbPhyHelper` classes to install those objects in the nodes created.

3. Configuration: Some important parameters are configured for the models of a protocol, which is incorporated by subscribing the collection of nodes' physical layers to a common channel

and then modifying the parameters of the channel like different data rates for uplink and downlink transmission.

4. Execution: The `Simulator` class traverse through all the scheduled events, after the simulation is started. During the simulation, the data are saved dynamically by the firing of trace sources.

5. Evaluation of performance: Post simulation the collected data is studied to validate the `cunb` module in Section 6.4.

4.3 C-UNB module

To model the behavior discussed in Section 2 of a C-UNB network, a new `cunb` module is created. This module comprises of classes that describe the behavior of C-UNB based smart meters and base stations at different layers, from the physical to the application layer. Figure 4.1 shows the list of important classes required for the simulation of the `cunb` stack on smart meters and base stations. Apart from the classes modeling a layer like the `CunbPhy` and the `CunbMac` in the `cunb` stack, additional classes are created to represent different functionalities like interference loss due to other packets using the same channel. This section further describes the overall architecture and interactions of the code, initiating from the application layers to the class modeled as the C-UNB channel. The detailed descriptions of the functions inside `cunb` module can be found in Appendix A.

The thesis work is primarily focused on the MS (Mobile Station or smart meter) and eNB (eNodeB or base station) implementations, since it is at this level that the C-UNB technique is incorporated.

4.3.1 Mobile autonomous reporting

The application layer class `MobileAutonomousReporting` is comprised of a packet generator which creates packets of a randomized payload size, based on the type of smart meters and their packet types. Since this application layer runs in the smart meter, we implement the DLMS-

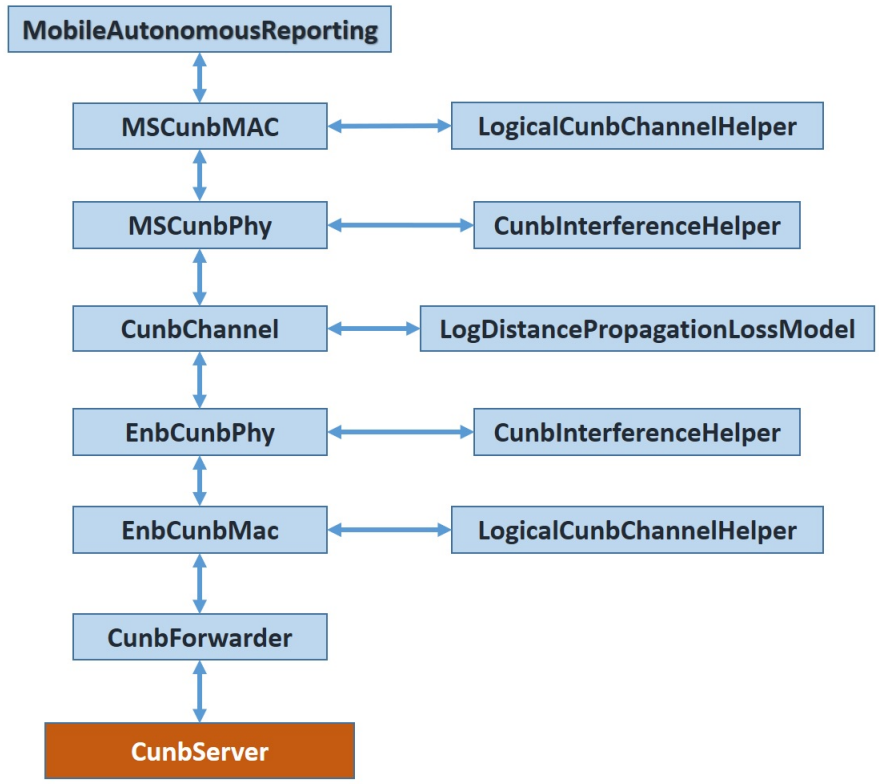

Figure 4.1: C-UNB module classes.

COSEM server, as explained in Section 1.6.1, inside this class. We set the reporting interval of 60 s for *KeepAlive* packets [4]. Similarly for the *Normal* and *Alarm* packets we followed other distribution. All xDLMS Application Protocol Data Units (APDUs) are put into the wrapper frame, prepended with the wrapper header. In the application layer, transmission refers to the forwarding of the packet to the lower layers after adding the upper layer headers. The application is first initiated on a smart meter with an random offset delay, determined by successful association of the smart meter with the DLMS-COSEM client, which is the C-UNB server or the data concentrator.

4.3.2 C-UNB MAC layer

The MAC layer of a C-UNB device is modeled in the `CunbMac` class. The objects of this class probes the micro-channels that are available through a `LogicalCunbChannelHelper` object.

Whenever a beacon packet is received, this class sets the network id, the cell id, the frequency and other parameters like the System Information Blocks (SIB) in LTE, based on the information collected from the beacons sent by the base station [5]. Since the C-UNB technology is device-triggered, as per [5], we also provide an alternative model where the mobile station initiates the communication using the `HelloSender` object where they communicate regarding the micro-channels they will be using for further transmission to the base station. Then, base stations are configured to receive the packets at that frequency for further communication with the mobile station.

Two subclasses `MSCunbMac` and `EnbCunbMac` are created that model the behaviors specific to a MS and an eNB or a base station.

4.3.2.1 Smart meter C-UNB MAC layer

The object of the `MSCunbMac` class characterizes smart meters' behavior. As this class controls the physical layers' state, it is essential to properly handle the state of radio, based on the status of the receive window. Algorithm 2 explains the procedure by which the MAC layer takes a packet from the transport layer and passes it to the physical layer with the addition of headers and trailers, and the selection of a random channel by the smart meter, as per the access control scheme mentioned in Section 2.2.3 for transmission. An event is scheduled in the `LogicalCunbChannelHelper` class for calculation of collision and interference.

4.3.2.2 Base station C-UNB MAC layer

The `EnbCunbMac` class differs from the `MSCunbMac` in that it simply implements the forward-only MAC layer. It forwards the data received from the smart meters to the C-UNB server and ACKs from the C-UNB server to the smart meters. This class stores the micro-channel information collected from the *Hello* packet sent by the mobile station. It uses the `PointToPointHelper` class of NS-3 to communicate with the C-UNB server in the Core/IP network.

Algorithm 2 Start of transmission method in the `MSCunbMac`.

Input: The `packet` containing the application and transport layer payload

1: Attach the link layer header. Set `segno` and `segcnt` attributes as per Figure 4.2.
2: Attach the frame header. Set the source MAC address.
3: Attach the MAC header. Set the `preamble`, `mtype`, `fsize`, `ack_flags` and `rep_cnt` for the header field. Set the `seq_cnt`, `ident` as per Figure 2.5.
4: Attach the MAC trailer. Set the `auth`, `fcs` and `ecc` as per Figure 2.5.
5: Assign each smart meter a uniformly chosen random location and eNB a fixed location.
6: Set the transmission parameters like the bitrate e.g., 250 bps in uplink.
7: Get the on air time to schedule an event in the `LogicalCunbChannelHelper` class.
8: Set the first and second re-transmit event.
9: Send the packet to the `MSCunbPhy` class.

4.3.3 C-UNB physical layer

The physical layer of a C-UNB device is modeled in the `CunbPhy` class. Particularly, this class replicates the behavior of the hardware in smart meters and base stations. When the smart meter requires to send a message, this class carries the MAC layer's packet and forwards it to the `CunbChannel` class. Then, it checks if the reception of a packet through the `CunbChannel` is error-free, based on the power of reception and the collision encountered by the packet. The class defines three possible states of a smart meter:

a) TX when a packet is transmitted;

b) RX when a packet is received;

c) IDLE when the smart meter is listening to the incoming requests or ACKs.

Like the `CunbMac` classes, `CunbPhy` also comprises of `MSCunbPhy` and `EnbCunbPhy` class, which represent in detail the physical layers of smart meters and base stations. Both objects

Algorithm 3 Reception of a packet in physical layers of smart meter and base station

Input: `packet`: this packet contains all the layer's header

Input: `sensitivity`: minimum power required to sense a packet at receiver in dBm

Input: `rxdBm`: signal's received power

Input: `d`: signal duration

Input: `freq`: signal frequency

1: Notify the `CunbInterferenceHelper` of the received signal
2: Drop the packet if in TX or RX state
3: Check for the `freq` if the node is a smart meter.
4: Check if the `rxdBm` is greater than `sensitivity`.
5: If all the conditions satisfy, switch to RX state and schedule reception of packet after `d` seconds.
6: After `d` seconds, check if the packet is destroyed by collision
7: If lost switch to IDLE mode from RX
8: If not lost forward to `CunbMac` class and switch to IDLE mode

calculate interference using `CunbInterferenceHelper` class that tracks signal that arrives at the node, and checks whether a packet encountered interference. Algorithm 3 shows the procedure followed by an `MSCunbPhy` object whenever the `CunbChannel` class notifies that a signal is arriving. Initially, it calls a `CunbInterferenceHelper` instance, which creates an event, that represents the signal, and stores the list of all events that arrived at that node. This event list contains data that is required for collision detection, such as the received power, the time at which the packet is received, and the micro-channel used. The received power of the signal is compared with the receivers' sensitivity and another reception completion event is scheduled if the packet can be successfully received. If the underlying physical layer is in IDLE state, then Algorithm 3 is performed because if a node is busy transmitting or receiving, reception of a new packet is impossible. Then, the `MSCunbPhy`'s `CunbInterferenceHelper` instance is used to check

if the current packet encountered collision. After getting the response, the node either forwards the packet to the upper layers or drops it.

The algorithm used to detect collision is described in Algorithm 4. The list of signals that interfered using the same micro-channels are grouped and their total energy is calculated. Interference energies are computed as the overlap time with the intended signal and the product of received power. After calculating the Signal-to-Noise and Interference Ratio (SNIR) values, they are compared with the sensitivity of the receiver to evaluate the collision of a packet.

Algorithm 4 Determine if a packet is lost due to collision.

Input: `packet`: the received packet
Input: `rxdBm`: the received power of the packet in dBm
Input: `d`: the received packet's duration
Input: `interferers`: list of interferers

1: `cumulativeInterferenceEnergy` = 0
2: for each interferer in `interferers`
3: Check if the frequency of interferers matches with packet
4: If the current interferer is older than 100 th event remove from the list
5: Get the overlap time between this event and the interferer
6: Calculate the interferer energy = interfererPower * overlap time
7: Calculate the current signal energy = `d` * current signal power
8: Calculate SNIR
9: If `snir` less than `snir_min` based on receiver sensitivity, packet is lost

In the `CunbPhy` classes we apply the packet tags. In NS-3, information regarding a packet can be stored in a packet tag, which is a customizable data structure [49]. We created a `CunbTag` class to store information about the transmission parameters that is used by a packet and the packet lost information. Hence, the information about a lost packet due to interference in the physical

layer is written to the `CunbTag` and the state of every packet in the simulation is informed to the simulation script.

4.3.4 C-UNB channel

The `CunbChannel` class models the wireless channel. This class takes the packet from the physical layer of mobile station and base station and delivers it to `CunbPhy` objects, with the computed received power as per the `LogDistancePropagationLoss` model of NS-3. While configuring the `CunbPhy` objects in the simulation script, it is bound to the channel. The `Send()` and `Receive()` functions are used by the physical layers to interconnect through the `CunbChannel` class. The physical layer calls the `Send()` method with parameters such as transmission power, duration of flight, frequency, and packet, when it needs to send a message through the channel. Then, the channel traverses the micro-channel list, and a `Receive` event is scheduled for those hosts listening to that micro-channel. For determining the time for scheduling a `Receive` event, the channel uses a `PropagationDelayModel`, as per the `MobilityModel` (i.e., location of nodes) of the base station and the smart meter.

4.3.5 C-UNB server

This class models the C-UNB server which acts as a data concentrator of the AMI network. This device does not use the MAC and physical layer of the `cunb` module, since it is located in the Core / IP network [5]. The primary job of this class is to de-duplicate the autonomous reports of the smart meters and to send the ACKs through the base station that received uplink transmissions with highest power. This class also hosts the DLMS-COSEM client, that sends AARQ and GETRQ requests to the server in the smart meter, as described in Section 1.6.1. This class mainly handles the objects of `MSStatus` and `EnbStatus` classes, whose functionality is described in 4.3.8. To communicate with the base stations, it uses the `PointToPointHelper` object of NS-3.

4.3.6 C-UNB net device

A "C-UNB network card" is modeled using the `CunbNetDevice` class. The applications of a `Node` use the `NetDevice` interface to forward packet to other C-UNB devices. An object of `CunbNetDevice` is utilized to bind the `cunb` objects that are attached in a node: a `CunbPhy` and a `CunbMac`. The `cunb` module utilizes the `NetDevice` that uses its `Send()` function to handle the MAC layer underneath.

4.3.7 Headers and trailers

There are two MAC header classes created for uplink and downlink purposes separately.

4.3.7.1 MAC header downlink

In C-UNB, the downlink MAC-PDUs (especially the ACKs) are optimized based on the fact that they are transmitted in response to the uplink transmission [5]. The downlink header is comprised of 56 bits which consists of the preamble for frame detection and bit rate synchronisation, a frame type, a payload length and acknowledgement bits.

4.3.7.2 MAC trailer downlink

The downlink trailer comprises of Authentication (Auth.) field, Frame Check Sequence (FCS) field and Error Correction Code (ECC) field. The authentication field is a 16-bit hash, computed using the Secured Hash Algorithms-1 (SHA-1) hash function of crypto++ library. It uses the 128-bit key and the device identifier, sequence number and the payload field of the uplink packet. When a downlink packet is triggered by a segmented uplink packet, its authentication field uses the last sequence number transmitted in the uplink. The frame check sequence in C-UNB downlink consists of a Cyclic Redundancy Check-8 (CRC-8). It checks the MAC-PDU after the error correction, before further decoding. Finally as per the specification [5], the ECC is generated using the Bose-Chaudhuri-Hocquenghem (BCH) code applied to the first four fields: header (excluding preamble), payload, authentication and FCS. The current version, does not implement the ECC

field. The complete MAC-PDU downlink is shown in the Figure 2.6.

4.3.7.3 MAC header uplink

In C-UNB, the uplink MAC-PDUs are triggered based on the reporting interval of application layer packet based on the packet type. The uplink header is comprised of 40 bits which consists of the preamble for frame detection and bit rate synchronisation, a frame type, a frame length, acknowledgement bits and a repetition counter. The repetition counter is set based on the retransmission count of the data packet.

4.3.7.4 MAC trailer uplink

The uplink trailer comprises of Auth., FCC and ECC field. These fields are computed the way the downlink trailer computes. Except in the uplink the ECC is generated using the Reed Solomon algorithm and is evaluated over the whole MAC-PDU except the preamble. The current version, does not implement the ECC field. The complete MAC-PDU uplink is shown in the Figure 2.5.

Figure 4.2: Link layer header in C-UNB uplink transmission. Adapted with permission from [5].

4.3.7.5 Link layer header

The C-IoT requirements define the packet size for the Mobile Autonomous Reports to be in the range of 20-200 byte length. As only small radio packets are affordable for the C-UNB radio access technology, it is proposed to have a segmentation/ re-assembly mechanism based on 31 blocks of data with a link layer header of 8 bits (4 bits for segment numbering and 4 bits for total

number of segments). Figure 4.2 shows the link layer header for the C-UNB uplink. In future version we will utilize this header to implement segmentation and re-assembly.

4.3.7.6 COSEM header

There are six types of application layer headers associated with DLMS-COSEM as described in Section 1.6.1, and they are AARE, AARQ, GETRQ, GETRE, RLRQ, and RLRE. In our module we have implemented the headers for first four types of packets, since we are evaluating the performance when the smart meters are associated with the data concentrator. The release packets are used to release an application association.

4.3.8 Smart meter and base station status

4.3.8.1 Smart meter status

The `MSStatus` class represents the C-UNB servers' knowledge about the smart meters in the C-UNB network it is controlling. The C-UNB server holds the information of each smart meter in the network. It holds the ACKs that the C-UNB server will send to this smart meter. It also keep track of all eNBs that are able to receive this smart meter's data. On new packet arrivals at the C-UNB server, the `UpdateEnbData()` method is called to update the mappings that associates an eNB's address to the received power it received the smart meters' last packet. This information is then used in the `GetSortedEnbAddresses()` method to return a list of the preferred eNB, based on the received power, to reply acknowledgements to this smart meter.

4.3.8.2 Base station status

The `EnbStatus` class represents the C-UNB servers' knowledge about the base stations connected in the C-UNB network. The server stores a list of instances of this class, one for each base station.

4.4 Helper classes

The helper classes in NS-3 are created to assist scripts in configuring the nodes to use the newly created module. Hence, apart from the core module explained in the above section, a collection of `helper` classes are designed to make the configuration of `cunb` network easier. For example, whenever an object of `HelloSender` application is instantiated in the script, its start time is set. For configuring multiple of such classes, the applications is started on a collection of nodes through the `HelloSenderHelper` class. Similarly, a few other helpers are created to correctly configure and install the C-UNB stack on the nodes. The classes `CunbHelper`, `CunbPhyHelper` and `CunbMacHelper` are developed to function in cooperation to create and incorporate `CunbNetDevice`, `CunbMac` and `CunbPhy` objects on the nodes, ensuring the configuration of each layer is done accurately for proper communication.

4.5 Other classes

There are more classes like `CunbForwarder`, `OneTimeReporting`, `BeaconSender`, `CunbInterferenceHelper` are created for different purposes. The functions associated with these classes can also be found in the Appendix A.

4.6 Summary

This version of `cunb` module successfully designs the MAC, physical and channels for the C-UNB technology. The major objectives of developing this module are to analyze the random access channel scheme and to evaluate the impact of the cooperative nature of multiple base stations, which were successfully tested in the current version. But the module need improvements in various areas such as the implementation of segmentation and reassembly in the link layer. Besides, the current version of `CunbNetDevice` does not support injection of real time traffic which can be extended.

5. IDS FOR THE AMI NETWORK[1]

5.1 Introduction

Based on time-criticality, the Smart Grid communication network can be classified into two types. The first one is the power transmission and distribution system where communication for monitoring, control and protection is time critical. For example, in case of IEC 61850 message, the message type Type 1A/P1, used for fault isolation have a time constraint of 3 ms [7]. The second one is the AMI network in which communication is time non-critical since it primarily involves interactions between customers and utilities. However the AMI network demands confidentiality and integrity. The protocols that are employed for AMI traffic are discussed in Section 1.6.

5.1.1 Attack scenario

A well-known security model widely used by many security experts, is the Confidentiality, Integrity, and Availability (CIA) triad, the three key principles which needs to be guaranteed in any kind of security system. The attacks are targeted mainly to disrupt these key principles of a secure system.

5.1.1.1 Attacks targeting availability

The Denial of Service (DoS) attacks can take place at different OSI layers in smart grid. At the physical layer, through jamming of channels in substations. In the MAC layer, an attacker can take advantage of the openness of the MAC address fields to masquerade itself as a genuine meter and perform Man-in-the-Middle (MiTM) attack. Similarly in the TCP/IP layer, DoS attack at both the layer can impair the end-to-end communication performance, such as distributed traffic flooding and worm propagation attacks [7]. At the application layer, DoS intend to exhaust the resources

[1]Parts of the material presented in this section are reprinted with permission from "Detection of rogue nodes in AMI networks" by A. Sahu, H. N. R. K. Tippanaboyana, L. Hefton and A. Goulart. *2017 19th International Conference on Intelligent System Application to Power Systems (ISAP)*, September 2017. ©[2017] by IEEE.

of hosts like CPU or I/O bandwidth. This attack has higher impact on the power distribution and transmission operation system in comparison to AMI network.

5.1.1.2 Attack targeting integrity and confidentiality

These attacks try to obtain and tamper the private information in smart grid. Example of an integrity attack is false data injection attack on electricity market to deliberately manipulate market price information. These attacks were earlier designed to impact the state estimation for the SCADA systems [7]. Eavesdropping communication channels using traffic analyzers and packet sniffers in power networks to obtain private information such as customer financial details, passwords and electricity usage are a few examples of attack on confidentiality. To prevent such confidentiality and integrity attacks, authentication and access control is highly essential.

For the AMI network it is essential to prevent attack against integrity and confidentiality because it would carry customer bank account information which should remain confidential and for the integrity part the information generated from the smart meter should not be masqueraded by an attacker.

5.1.2 Motivation for ARP spoof detection

The NAN of AMI is generally a fixed wireless network, where the data concentrator and the smart meter devices are not mobile. Like any other wireless networks, fixed wireless networks possess many security challenges, and the presence of an attacker within the range of the network makes it a vulnerable environment. Among the attacks, MitM attack is a dangerous and easily executable attack. In our work, we have implemented it using an open source tool called Ettercap. A MitM is an active eavesdropping attack, where the attacker intercepts communication between two legitimate hosts. It is carried out on the MAC layer through ARP [50] cache poisoning. This means that the attacker receives packets from both smart meter and data concentrator. After the interception of connection between the legitimate hosts, the attacker captures traffic, reroutes them to unknown hosts, injects malicious code and data, or advertises flawed services to neighboring

meters.

The ease with which MitM attack can be implemented makes it very important to detect and stop it earlier. This is the main reason that motivated us to develop an Intrusion Detection System (IDS) for MitM attacks in AMI networks. An IDS is a software that monitors incoming data packets to a single computer or a network of computers to identify malicious activities. Our contribution is to present the design and preliminary results, obtained in a real-time simulation testbed, of a host-based IDS that protects the smart meter and a network-based IDS that protects the data concentrator in the NAN.

5.2 Host-based IDS for ARP spoof detection

A host-based IDS analyzes the traffic to and from a specific host on which the IDS is installed. It is usually deployed in the scenarios where the network-based IDS is not sufficient to handle traffic for large network. Since the AMI network comprises of a huge number of smart meters we decided to design host-based IDS.

5.2.1 Feature vectors for ARP attack detection

For the detection of the ARP spoof attack we have considered two features of the packet received at the smart meter. The first one being the Round Trip Time (RTT) from the ACK received from the data collection server and the second one is the occurrence of frequent gratuitous ARP. We do not consider the global IP-MAC mapping cache technique to detect attacker because it is not feasible for a smart meter to record so many smart meters.

5.2.1.1 Round Trip Time (RTT)

RTT is the time taken for a packet once it leaves the interface of the source to reach the destination and its corresponding ACK received by the source. In Figure 5.1 we can observe that due to MitM attack the round trip time is increased. In this case the RTT is the sum of the transmission, propagation delay, and processing delay at the nodes. When an attack is taking place the processing time at the attacker, i.e., mBA, increases for each packet. A similar approach using RTT was

adopted in [51]; however, RTT can increase due to network congestion. Thus, we improved it by adding a second feature, which is described next.

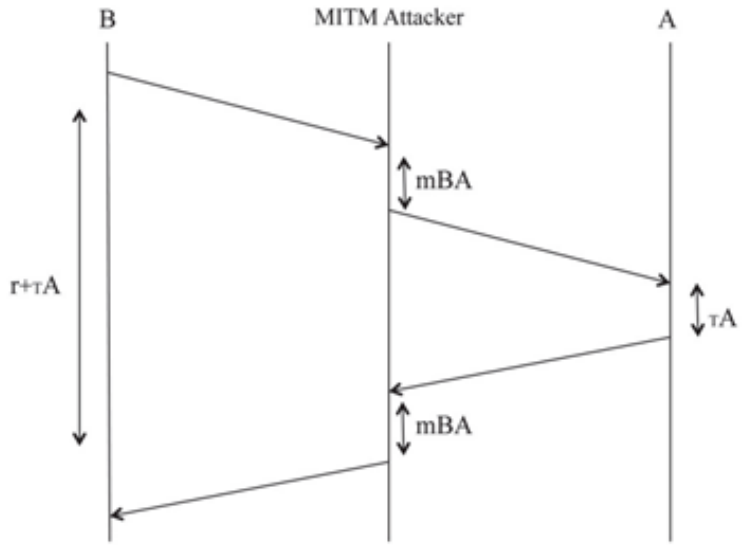

Figure 5.1: Increase in Round Trip Time under attack. Reprinted with permission from [2].

5.2.1.2 Gratuitous ARP

When we receive an ARP reply without any ARP request, then such ARP packets are called gratuitous ARP packets. Usually, these gratuitous ARP packets do not come continuously. For the victims' ARP cache to be poisoned during the time of attack, the attacker needs to send the gratuitous ARP packets continuously. In Figure 5.2 we can see a gratuitous ARP detected by wireshark. We consider this fact also along with RTT to design our IDS.

```
378 163.367182   10.1.2.1          10.1.2.20         ICMP      60 Echo (ping) request  id=0x7ee7, seq=32487/59262, ttl=64 (reply in 379)
379 163.367412   10.1.2.20         10.1.2.1          ICMP      42 Echo (ping) reply    id=0x7ee7, seq=32487/59262, ttl=64 (request in 378)
380 163.367481   Universa_6b:b5:d9 CadmusCo_ff:8c:51  ARP       60 10.1.2.1 is at 00:24:7e:6b:b5:d9
381 163.370830   00:00:00_00:00:03 Broadcast         ARP       60 Who has 10.1.2.20? Tell 10.1.2.1 (duplicate use of 10.1.2.1 detected!)
382 163.370891   CadmusCo_ff:8c:51 00:00:00_00:00:03 ARP       42 10.1.2.20 is at 08:00:27:ff:8c:51 (duplicate use of 10.1.2.1 detected!)
```

Figure 5.2: Gratuitous ARP packet detected by Wireshark. Reprinted with permission from [2].

5.3 Bayesian approach to ARP spoof detection at smart meters

As discussed in Section 5.2.1, the RTT and the observation of continuous gratuitous ARP can help detect an attack. But sometimes due to transmission errors and network congestion in the network or change of meter configuration, prediction techniques may give rise to false positives. In our detection model, we propose a Bayesian based iterative algorithm.

5.3.1 Posterior probability prediction algorithm

The smart meter usually communicates with the HES through the data concentrator. The meter continuously probes the ACKs, and observes the RTT delays. As the meters are fixed, the RTT delays are not going to vary much until and unless there is disturbance in the network. Thus, let us define the prior probability of the meter under attack as $P(A)$ and not under attack as $P(A')$, where $P(A)$ and $P(A')$ satisfy Equation 5.1.

$$P(A) + P(A') = 1 \tag{5.1}$$

The observed features RTT and gratuitous ARP can be denoted as T_r and T_g respectively. The conditional probability or the likelihood of the features can be expressed as $P(T_r, T_g|A)$ and $P(T_r, T_g|A')$.

Whenever the meter observes the occurrence of features T_r and T_g, the posterior probability, $P(A|T_r, T_g)$ is computed using the Bayesian theorem as per Equation 5.2.

$$P(A|T_r, T_g) = \frac{P(A)P(T_r, T_g|A)}{P(A)P(T_r, T_g|A) + P(A')P(T_r, T_g|A')} \tag{5.2}$$

If the IDS decision about whether an attack is detected depends only on $P(A|T_r, T_g)$, it is highly probable that the decision made is inaccurate. Further it will impact the decision of upcoming packets. To handle this shortcoming, we probe the features from consecutive packets denoted by C, denoted as $C = \{C_i\}, C_i \in T, t \leq i \leq t + a$. Further, we consider the probability $P(C|A)$ to

calculate the likelihood $P(A|C)$. Detection with more features from a collection of packets will give more accurate response. Consequently, we take the posterior probability $P(A_{t+a}|T_r, T_g)$ as the correction prior probability for $P(A_t)$. It is calculated by using the iterative prediction as per Equation 5.3.

$$P(A_{t+1}) = P(A_t|T_r, T_g) = \frac{P(A_t)P(T_r, T_g|A_t)}{P(A_t)P(T_r, T_g|A) + P(A'_t)P(T_r, T_g|A'_t)} \tag{5.3}$$

5.3.2 Attacker detection algorithm

When the smart meter upgrades its $P(A)$, it is possible to decide whether there was an attack based on upgraded $P(A)$ as per the inequality in Equation 5.4:

$$P(A) > P_t \tag{5.4}$$

where, P_t is the threshold, and $0 \leq P_t \leq 1$. Its value has an influence on misjudgment due to latency due to network congestion. To prevent these erroneous detection, a procedure to modify the threshold dynamically is required [33]. In this thesis, we probe the average number of abnormal packet flagged in a time duration denoted by α and average abnormal packet confirmed as an attack packet by β. We then adjust the value of P_t according to Equation 5.5:

$$P_t = \begin{cases} 1 & 1 \leq P_t + (\alpha - \beta) * \mu \\ P_t + (\alpha - \beta) * \mu & P_{\text{min}} < P_t + (\alpha - \beta) * \mu < 1 \\ P_{\text{min}} & P_t + (\alpha - \beta) * \mu \leq P_{\text{min}} \end{cases} \tag{5.5}$$

where, μ is the step size for learning and P_{min} is the lower limit of P_t. If β is less than α, P_t is increased to reduce misjudgement. Similarly, if α is less than β, we decrease the P_t to detect the attacker rapidly.

5.4 Network-based IDS for the data aggregator

In AMI networks, smart meters have three characteristics that make security a difficult task: limited storage, limited processing capacity, and consideration of meters to be trustworthy. Because of their limited storage and processing capacity, smart meters do not support complex IDS systems, even if smaller host-based IDS are installed on them. Thus, there must be IDS systems running at a higher level in the network, such as at the data aggregator/concentrator level. Furthermore, rather than relying on pre-built IDS systems, we chose to build an example program that detects only one type of attack as proof of concept for several reasons:

1. Pre-built systems are rarely lightweight and unsuited to the low processing capacity environment. Even at the collector level, running complicated systems built to handle all types of traffic is unnecessary.

2. Pre-built systems are expensive or difficult to use, and sometimes not built to work with smart grid protocols. For example, Snort, though open-source requires payment to be deployed in a development environment. Furthermore, writing rules that do not exist for certain types of attacks can be time consuming and requires developers with intimate knowledge of Snort rule systems, preprocessors, and logging frameworks. BRO is another popular alternative but was found to be unsuited to analyzing protocols below Layer 4, and was primarily suited to TCP-based protocol analysis.

3. AMI networks only carry a few protocols. Thus, considering the complexity of pre-built systems and the relative simplicity of a custom solution, development costs and performance both stand to benefit from lightweight systems. Such systems focus on AMI traffic in particular and log only certain types of attacks.

The type of attack we chose for our proof-of-concept system was an ARP cache poisoning, also known as ARP spoofing. Most ARP spoofing detection techniques often fail to recognize

malicious traffic or they classify non-malicious traffic as malicious. Thus, we have several boolean checks and a number of structures that track different nodes and activity. They are optimized to reduce redundancy and empty old response caches after certain time periods.

These are the structures for ARP spoofing detection: an ARP cache like any other, generated from any ARP packet; a cache generated only from ARP requests, since spoofing using requests would not do attackers any good (responses would just go to the wrong place); a cache of the most recent ARP requests (`RecentRequests`); information about each smart meter (`MeterInfo`), organized as a dictionary of MAC-IP address pairs, the number of gratuitous responses it has sent, and the timestamp of its last gratuitous response; and a similar dictionary for meters tagged as malicious (`AttackMeterInfo`), which also contains a list of victim IP and MAC addresses.

As shown in Figure 5.3, an ARP request simply updates the `Safe Cache` and gets marked in the `RecentRequests` cache. ARP responses go through a number of checks. First, the latest-request cache is searched for a request within the last 10 s and if it is found (and not expired), it moves on; otherwise, it is tagged as gratuitous. At this time, the `RecentRequests` table is updated, since it is the only opportunity to check the table without causing extra overhead. A timestamp is registered for this packet and compared to the last timestamp: if it is more than 30 s after another is received, the number of gratuitous responses is decremented by 1 per 30 seconds before adding to its number. Then, the cache is updated with the MAC-IP mapping of the packet. If the mapping is different than the one in the `Safe Cache`, its IP has changed. The list of IP addresses associated with the MAC in the `MeterInfo` is updated; then, the number of gratuitous responses it has sent is checked for being greater than one. If both conditions are met, it is added to the `AttackMeterInfo` table. Its original IP is noted as its IP and the IP in the response packet is logged as a victim.

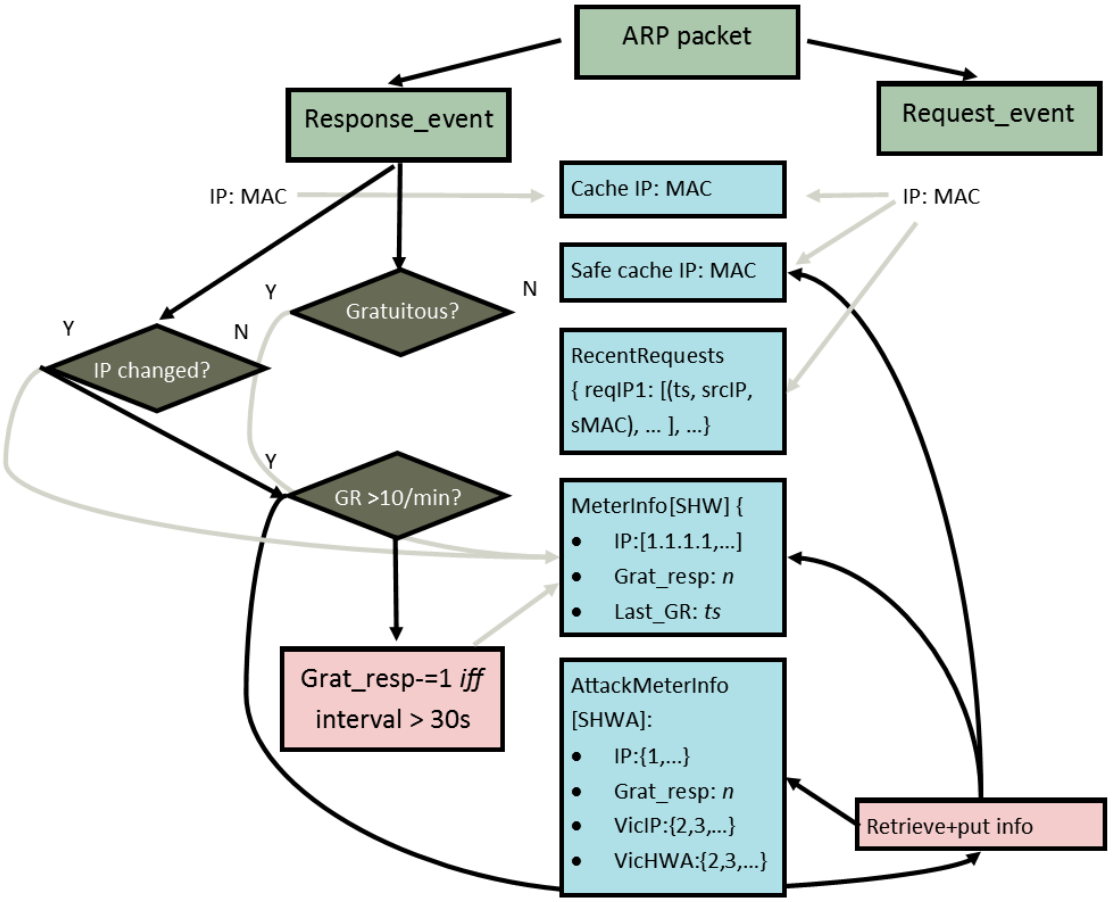

Figure 5.3: Intrusion detection logic for network-based IDS. Reprinted with permission from [2].

5.5 Summary

AMI networks carry many important information and need to be secure. In this section, we have used RTT delays and frequency of Gratuitous ARP features to design two IDSes: a Host-based IDS to detect ARP spoofing attack at the smart meters and a Network-based IDS for the data aggregator. We implemented the host-based IDS on our testbed and in Section 6.5 we will evaluate and discuss the performance.

6. SIMULATIONS, RESULTS AND ANALYSIS[1]

6.1 Introduction

This section analyzes the results obtained from different simulations. We have divided this section into four subsections. The first subsection discusses the performance evaluation of C-UNB using SimPy simulator. The second presents the comparative study of different wireless technologies performed in the NS-3 based real-time simulation testbed. The third validates the functionalities incorporated in the new `cunb` module in NS-3. The validation process includes performance metrics like throughput and retransmission probability. The final one shows some preliminary results on the proposed IDS for ARP spoof detection.

6.2 C-UNB performance analysis using SimPy python simulator

The goal of our experiments is to study the impact of the C-UNBs' random access control scheme for large number of smart devices generating data at a different reporting interval in a single cell.

6.2.1 Assumptions

For house-to-grid communications, many types of applications are being planned, such as monitoring, billing, and demand response. The latter includes real-time pricing and some forms of auction or bidding. In this simulation, we assume that monitoring means that the smart device at the customers' home sends reports to the data collection server at the utility company. Such reports can be periodic measurement reports or exception reports such as alarms (e.g., power failure). The

[1]Parts of the material presented in this section are reprinted with permission from "Cellular IoT for Mobile Autonomous Reporting in the Smart Grid" by A. Goulart and A. Sahu. *International Journal of Interdisciplinary Telecommunications and Networking*, July 2016. ©[2016] by IJITN,
"Modeling AMI network for real-time simulation in NS-3" by A. Sahu, A. Goulart and K. Butler-Purry. *2016 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, October 2016. ©[2016] by IEEE, and
"Detection of rogue nodes in AMI networks" by A. Sahu, H. N. R. K. Tippanaboyana, L. Hefton and A. Goulart. *2017 19th International Conference on Intelligent System Application to Power Systems (ISAP)*, September 2017. ©[2017] by IEEE.

Table 6.1: Types of AMI traffic. Reprinted with permission from [4].

Traffic	Report Interval	Avg Packet Size
Periodic keep-alive requests	1 min	92 bytes
Periodic AMI request/response	240 min	254 bytes
Aperiodic traffic	N.A.	800 bytes

traffic for application layer protocol used for smart meter communication is classified into three types of packets as per the data collected in [35]:

- Periodic keep-alive requests
- Periodic AMI request/response
- Aperiodic traffic

The periodicity and the packet size of different AMI traffics are detailed in Table 6.1.

The C-UNB simulation experiments used the following assumptions:

1. The MAC layer of smart meters has no access control delays (grant-free access). If a device has data to send, it transmitted immediately as in ALOHA networks. Each smart meter chooses a micro-channel at random. It uses the same channel for the uplink transmission of the packet and its acknowledgement.
2. The propagation delays in the uplink and downlink are assumed to be zero. The total delay is considered as the sum of transmission time plus the loop-back time between base station and C-UNB server, which is taken in the range of 4 to 6 s and it includes the processing time at the server. The processing time is the time spent in the de-duplication of data and the selection of base station process to send the ACKs.
3. Beacon channel packets are not used as we only considered single cell.

4. According to Section 2, the effective bandwidth of the channel is considered to be 200 kHz and of each micro-channel to be 500 Hz. There can be a total of 360 micro-channels (180 kHz / 500 Hz). We scaled it down to eight channels in our simulation.

5. It was assumed in [16] to have 4500 smart devices in each cell. Using our scaling factor of 45, we considered 100 smart meters for the simulation.

6. We have considered only two base stations for the simulation.

7. Among all the devices, 90 percent of them are household devices and 10 percent are commercial devices which generated packets of larger size.

8. Commercial and household packets used separate channels for transmission. Out of eight channels, one channel is dedicated for commercial packets.

9. Alarm packets' arrival rate followed an exponential distribution with a mean of one hour. Keep-alive were periodic with the periodicity of 60 s and AMI (normal) packets with the periodicity of four hours.

10. For simplicity, we did not consider fragmentation. The total packet size was sent as a single packet, including multiple headers used for each fragment.

11. For household smart meters:

    - Keep-alive packet:

      Single packet size is 141.5 bytes.

      Payload is 92 bytes.

      Each packet header size is 16.5 bytes.

      As we are not taking segmentation, packet size is 141.5 bytes (92 + 3*16.5).

    - AMI Request/Response (normal) packet:

Single packet size is 386 bytes.

Payload is 254 bytes.

Each packet header size is 16.5 bytes.

As we are not taking segmentation packet size is 386 bytes (254 + 8*16.5).

- Alarm packet:

  Size is exponentially distributed with a mean of 200 bytes.

12. For commercial smart meters:

    These packets are double the size of household packets.

    Keep-alive packets' size is 283 bytes.

    AMI request/response is 772 bytes.

    Alarm packets' size is exponentially distributed with a mean of 400 bytes.

13. A log-normal shadowing propagation model as per Equation 3.1 was considered to calculate the probability of each base station receiving a packet from a device, depending on their distance to base station and received power levels. The path loss exponent for the urban environment was considered to be 3.5. The minimum required received power at base station was assumed to be -120 dBm.

14. A random back-off value has been considered in the case where there are retransmissions.

15. The maximum number of retransmissions for alarm packets are three, normal packets are two and no retransmissions for keep-alive packets.

16. The timeout value was considered to be twice the Round Trip Time (RTT).

6.2.2 Simulation set-up

We implemented the simulation set-up using the SimPy simulation [52], which is a process-based DES framework based on standard Python. The Python generators are utilized to dispatch event and is usually used for asynchronous networking.

1. Maximum of 300 smart meters are considered. We tested the performance with varying amount of smart meters.

2. The sampling interval at the smart meter is considered to be 0.2 s. For example, a packet reaches at time t=14.56 s, then the meter would respond at time t=14.6 s. If we increase the sampling frequency the computation time for the simulation increases considerably.

3. Two threshold values of probability for successful reception (i.e., the probability that the received power at base station is greater than -120 dBm) are considered for the base stations, i.e., 0.75 and 0.9.

6.2.3 Results and analysis

1. We compared the impact of number of base stations on the successful transmissions. It can be observed from Figure 6.1 that with the usage of more base stations the amount of packets that are successfully transmitted are higher than the case with one base station.

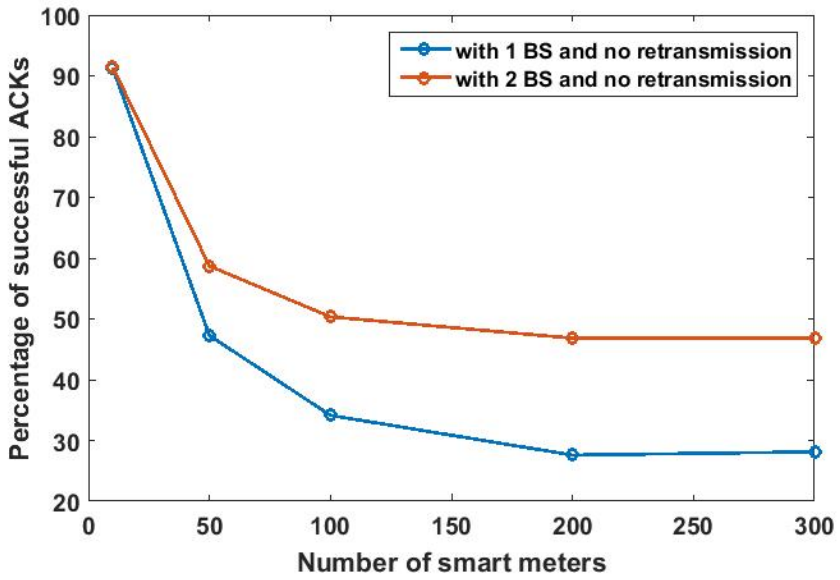

Figure 6.1: Impact of number of base stations on successful transmissions.

2. Similarly from Figure 6.2 it can be observed that the amount of packets encountering collision is higher with the usage of one base station. With one base station the percentage of collision is about 70 percent, which is more than three times than that with two base stations.

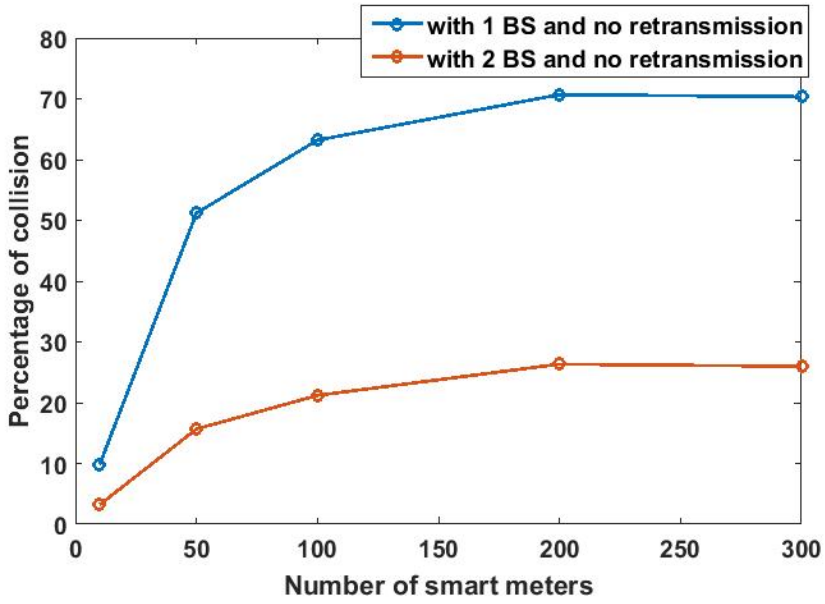

Figure 6.2: Impact of number of base stations on collision rate.

3. We also studied the effect of probability of reception threshold (`Pthres`) on the packets that are successfully acknowledged. Figure 6.3 shows that with the increase in threshold probability the amount of packets successfully acknowledged increases. Threshold probability for successful reception of value 0.9 performed better than 0.75. It can be concluded that lower the threshold, more number of packets are received at the base station, resulting in higher collision rates.

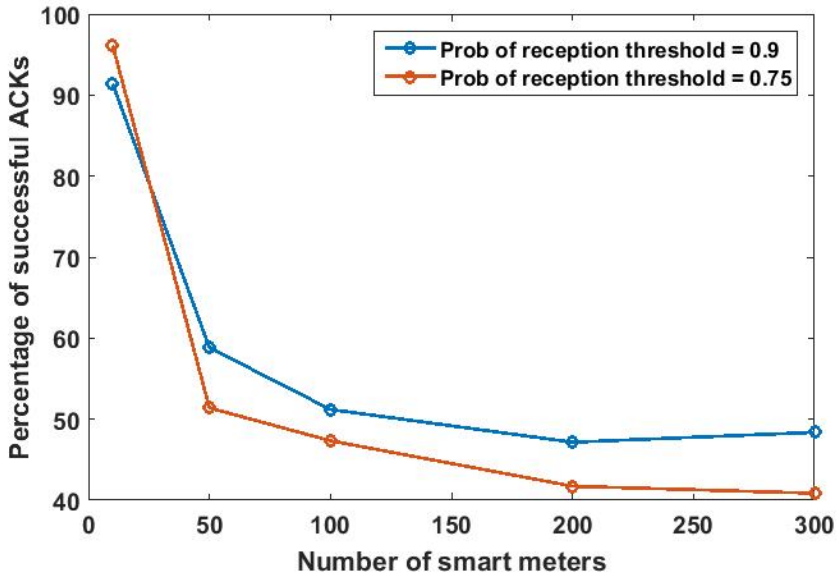

Figure 6.3: Effect of `Pthres` on packets that are successfully acknowledged.

4. With the implementation of retransmission the average delay of a successfully acknowledged packet increased. As shown in Figure 6.4, the average delay increased from 16-17 s without any retransmissions to 35-40 s with retransmissions.

5. Using retransmission the amount of packets that are successfully acknowledged increased. As it can be observed from the Figure 6.5 almost all the packets are successfully transmitted due to retransmission. Without retransmission only 50 percent of packets are successfully acknowledged when more than 200 smart meters are used. With a trade-off of delay we get more percentage of successful acknowledgments by using retransmission.

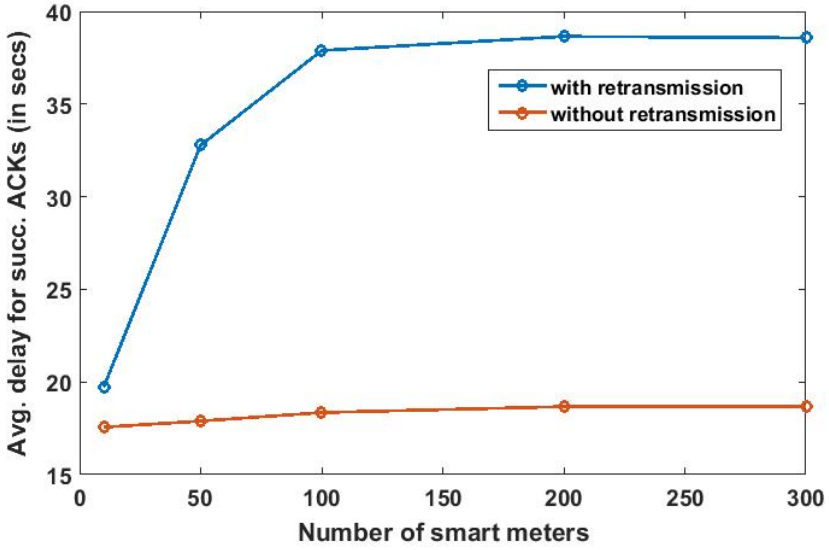

Figure 6.4: Effect of retransmission on average delay.

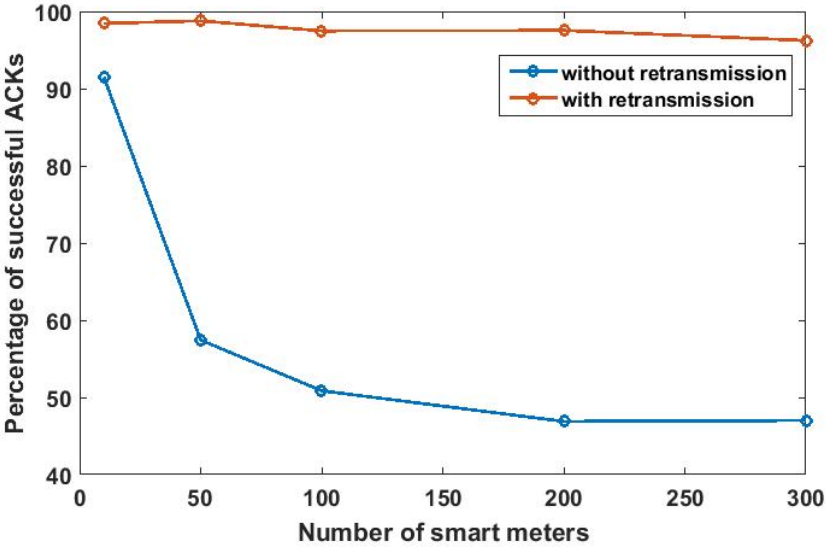

Figure 6.5: Effect of retransmission on successful acknowledgements.

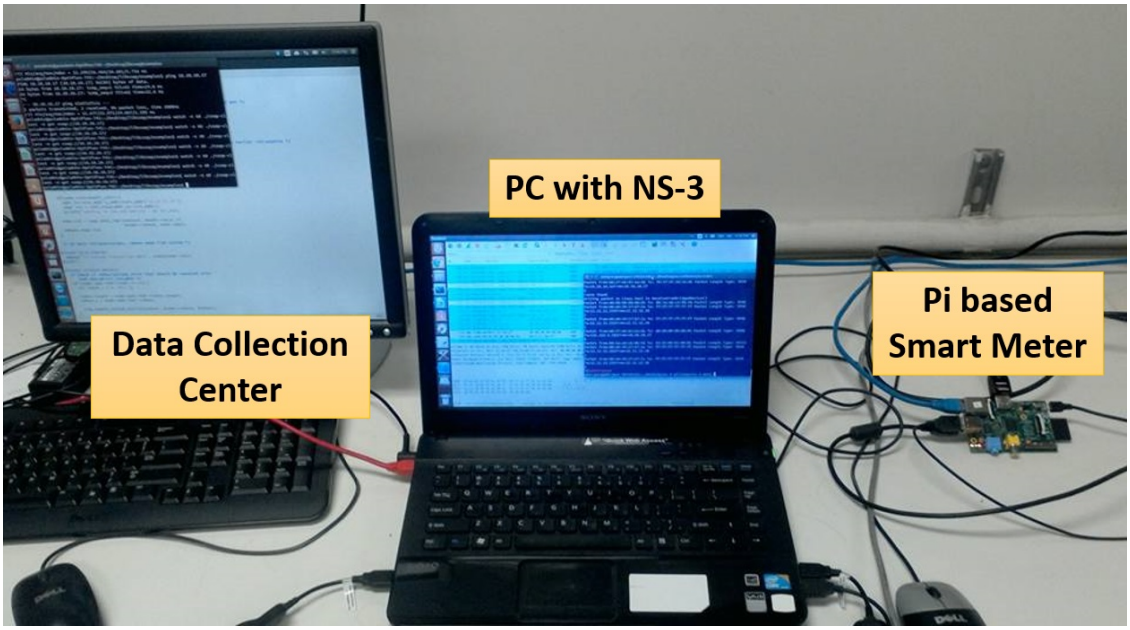

Figure 6.6: Testbed setup for AMI network. Reprinted with permission from [6].

6.3 NS-3 real-time simulation

6.3.1 Simulation set-up

Our testbed consists of two computers and a Raspberry Pi as shown in Figure 6.6. The desktop computer in the left emulates a data collection center where Constrained Application Protocol (CoAP) and HTTP clients are running on Linux operating system. The Raspberry Pi acts as a smart meter, for it generates AMI traffic based on the parameters in Table 6.1. The laptop in the middle runs the NS-3 simulation and wireless models. It has an Intel i3 processor, 4 GB of RAM, and Ubuntu 14.04. It captures real-time packets through two ethernet ports: one ethernet port is connected to the desktop computer, the other is connected to the Raspberry Pi.

6.3.1.1 Smart meter traffic

We assumed three modes of AMI traffic: fixed-scheduling, event-driven, and demand-driven [11]. In fixed scheduling, a smart meter reports data periodically. In event-driven, data is generated based

on an event at smart meters; for example, when electric consumption crosses a threshold value, it sends an alarm packet. In demand-driven mode, the utilities poll for faults and current state of smart meters. The event-driven mode has higher priority than the other modes. In the simulations performed in the Section 6.2 of AMI network using C-IoT, we tested periodic AMI request/response traffic, using different report intervals, which can be classified as fixed-scheduling. In this simulation, we consider the demand-driven mode where the data collection center acts as a client polling for the state of smart meters, which act as servers. Both CoAP and HTTP application-layer protocols are used to emulate demand-driven smart meter traffic.

1. CoAP: It is a light-weight application protocol for nodes with low computational capacity and power such as smart meters and IoT devices [53]. To facilitate its integration in small devices, we tested CoAP with UDP protocol and four-byte headers, and we used the libcoap [54] libraries. The most basic case of piggybacked response is implemented, where the response carried the acknowledgments for the request.

2. THTTP: We used thttpd (tiny /turbo /throttling HTTP) [55] to explore the effect of background traffic on connection establishment time. This lighter TCP-based HTTP protocol is used in LTE smart meters inside NS-3. Similarly, a lightweight (lighthttpd) web server is used in the Raspberry Pi, emulating the Wi-fi and WiMAX enabled smart meters.

6.3.1.2 SYN flood attack model

Our testbed also allows us to test various cyber attack scenarios and explore network performance during the attacks. A simple SYN flood attack is explored on the WiMAX based AMI network. A SYN flood is a type of DoS attack, where an attacker sends a succession of TCP SYN messages to a victims' system to make it unresponsive to legitimate traffic. The network model for this attack is shown in Figure 6.7, where a third computer is used as the backup data collection center. Three TAP bridges are used to connect client, server and backup server to the NS-3 network simulator.

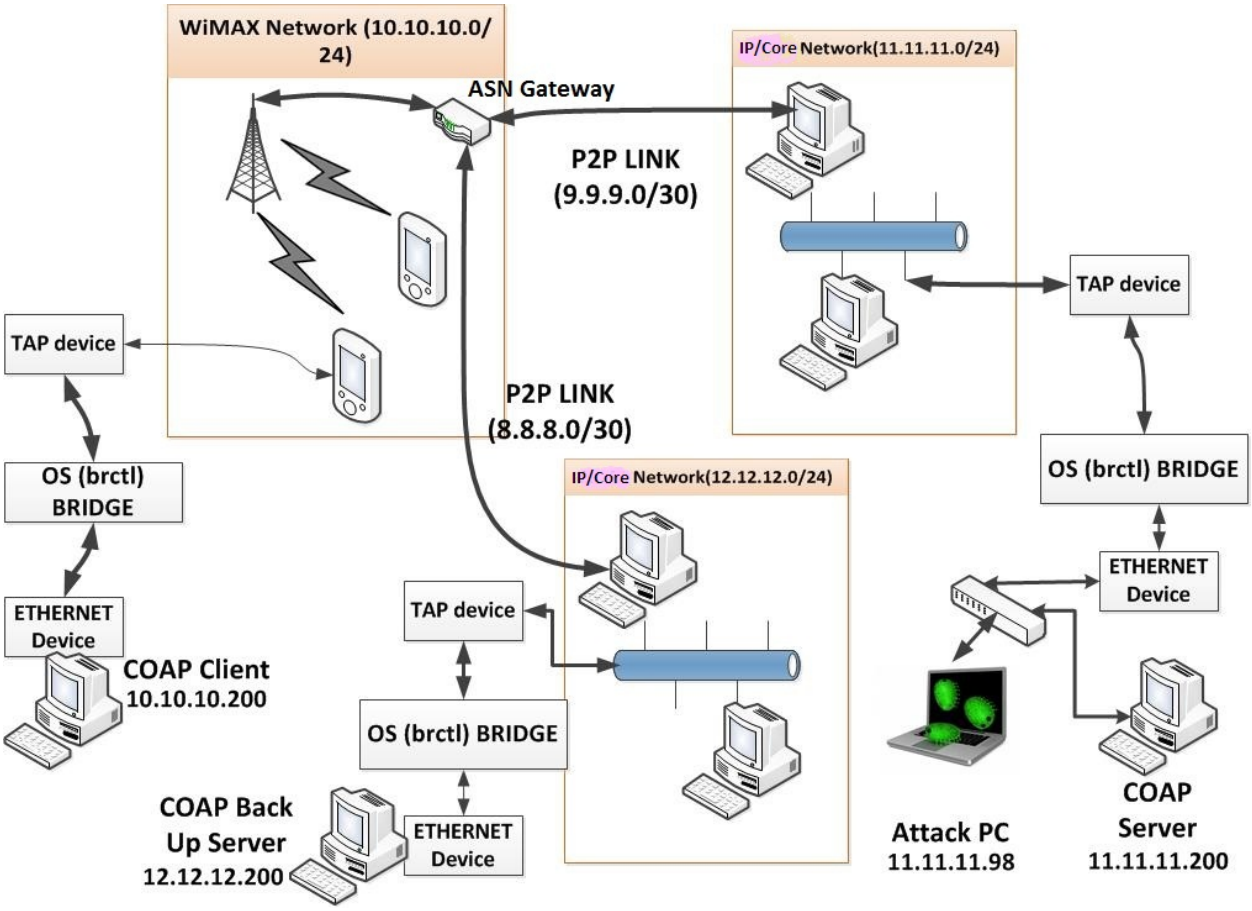

Figure 6.7: SYN flood attack model.

6.3.2 Results and analysis

6.3.2.1 Wi-Fi simulation

In this simulation, we analyze the impact of background traffic on the TCP connection establishment and release time for HTTP traffic. Wi-Fi stations (STA) generate background traffic by sending exponentially distributed random-sized UDP packets with mean size of 800 bytes and a Poisson distributed arrival interval with mean 2 s. The HTTP client generates the AMI request at a constant interval of 1 min for 10 mins. The AMI response for each request is a 1071-byte HTML file. We increased the number of background nodes by 2 (a sender-receiver pair) from 0 to 12

nodes. As shown in Figure 6.8, the average TCP connection time increases, because of retransmission of ACKs and SYN-ACKs which collide with background traffic. The bars in the plot indicate the 95 percent confidence interval. When we reduced the report interval from 1 min to 10 s, some packets do not even release the connection completely.

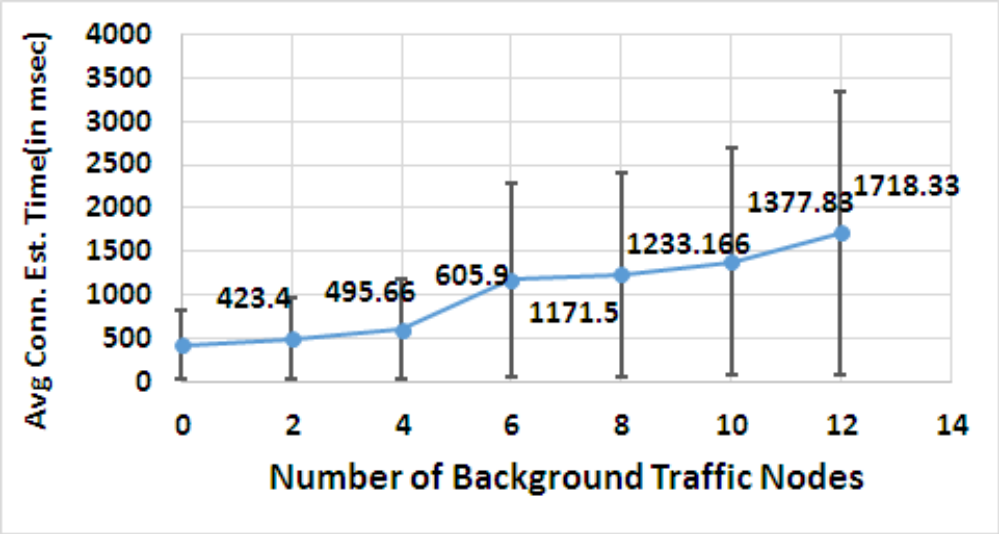

Figure 6.8: Average TCP connection establishment time for IEEE 802.11ah with HTTP traffic. Reprinted with permission from [6].

The average throughput for the smart meter traffic is calculated using transmission time of HTTP packets, as per the Equation 6.1.

$$\text{Average throughput} = \frac{\text{Total data payload}}{\text{Total transmission time}} \tag{6.1}$$

As we can see from Figure 6.9, the throughput degraded with the increase of background traffic. The average throughput reduced from 18.68 kbps to 3.358 kbps when the number of background nodes was increased to 12.

An IoT-based CoAP protocol is also used to evaluate the request/response time with the same

reporting interval of 1 min for a duration of one hour. The average time taken for the CoAP client to receive the 300-byte response from CoAP server at the Raspberry Pi, under varying background traffic, is shown in Figure 6.10. Note that a few packets experience a response time as high as 7 s when the number of background nodes was increased to 12. Such a large variance in response time is due to alternating high (7 s) and low (70 msec) response duration.

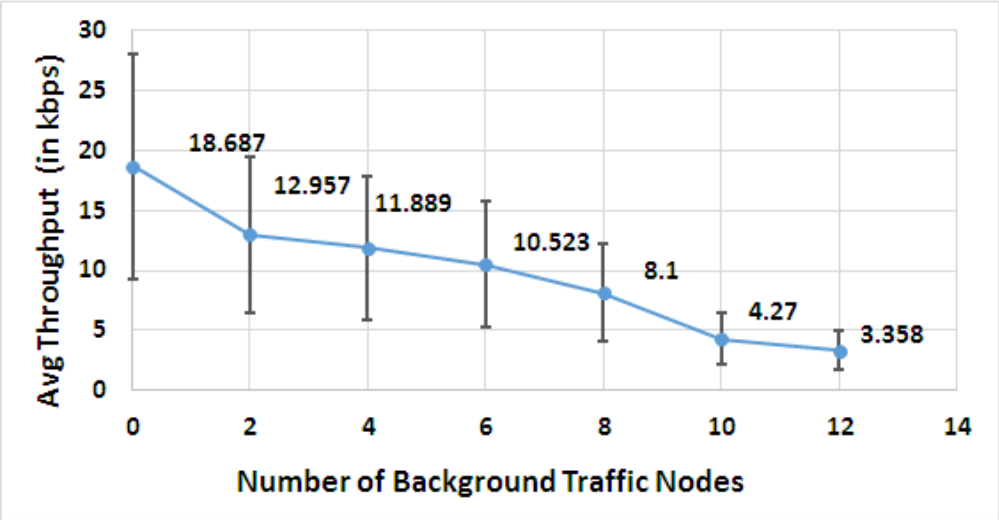

Figure 6.9: Average throughput for IEEE 802.11ah with HTTP. Reprinted with permission from [6].

The default maximum number of retransmissions for a CoAP packet is four. We then calculated the number of requests, responses, and retransmissions in Figure 6.11. The number of packets experiencing retransmission in case of 6 background nodes is almost three times that of zero background traffic.

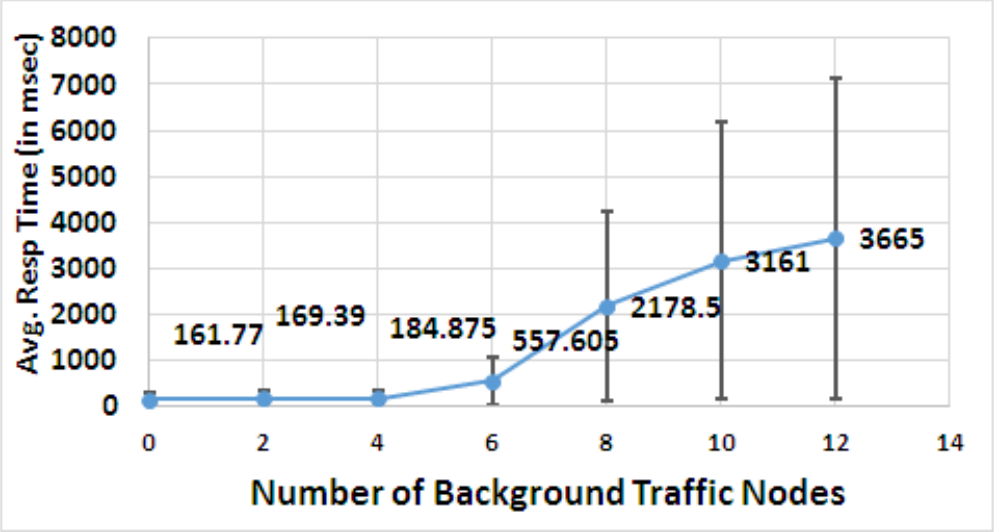

Figure 6.10: Average request/response time for IEEE 802.11ah using CoAP. Reprinted with permission from [6].

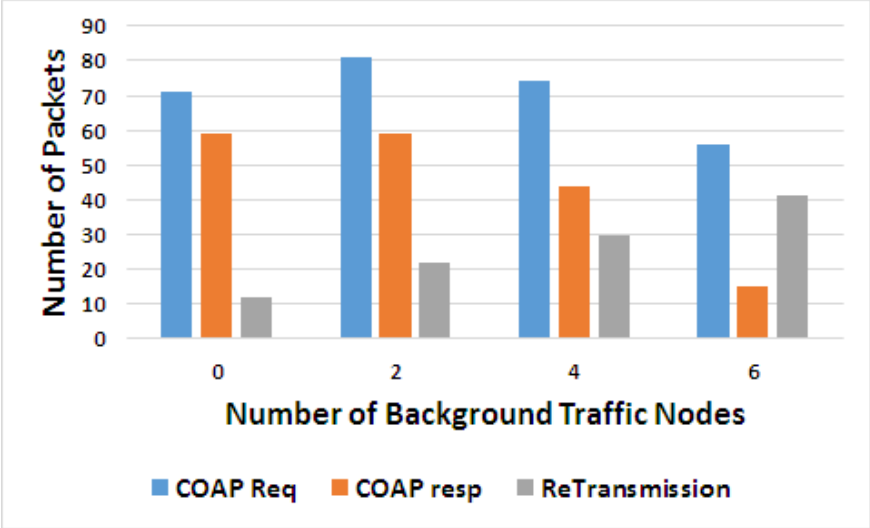

Figure 6.11: Request, response, and retransmitted packet distribution of CoAP packet for 802.11ah model. Reprinted with permission from [6].

6.3.2.2 WiMAX simulation

In this scenario, we increase the background traffic and analyze the TCP connection establishment and release time using HTTP protocol with a payload of 1071 bytes. The background traffic packet size and arrival interval follows the same distribution as that of the Wi-Fi simulation. The results are shown in Figures 6.12 and 6.13. The average connection delay for WiMAX is almost three times lower than 802.11ah, as WiMAX uses higher data rate of 27.65 Mbps. As the background traffic increases to 12 nodes the connection establishment and release time goes beyond 350 msec.

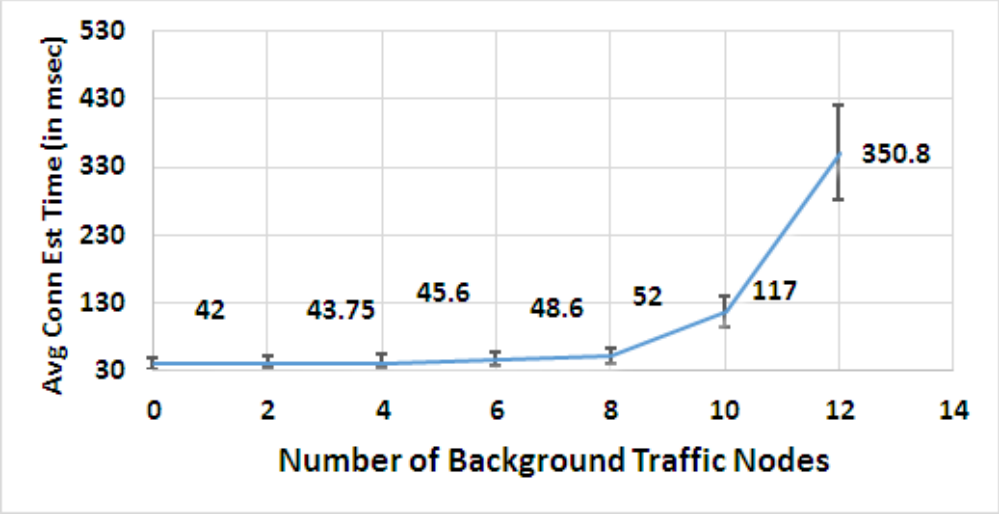

Figure 6.12: Average connection establishment time for WiMAX. Reprinted with permission from [6].

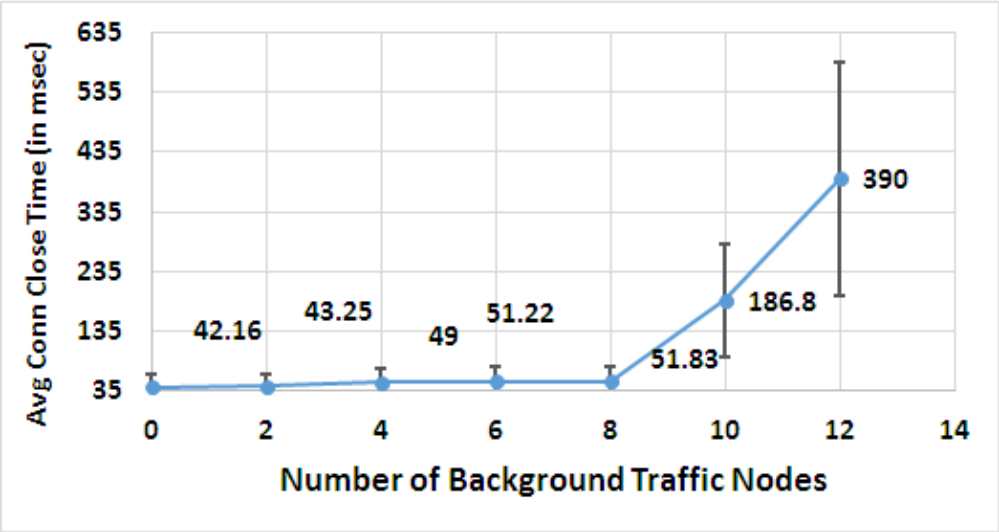

Figure 6.13: Average connection closing time for WiMAX. Reprinted with permission from [6].

From Figure 6.14, the throughput degradation with the increase of background traffic is observed. The average throughput reduced from 18.68 kbps to 3.358 kbps when the number of background nodes was increased to 12. Using CoAP protocol, we evaluated the average response time by modifying the background traffic for 300-byte AMI payloads as shown in Figure 6.15. The average response time for the CoAP packets too degraded with 12 background nodes. Therefore, we have restricted our simulations to 12 background nodes.

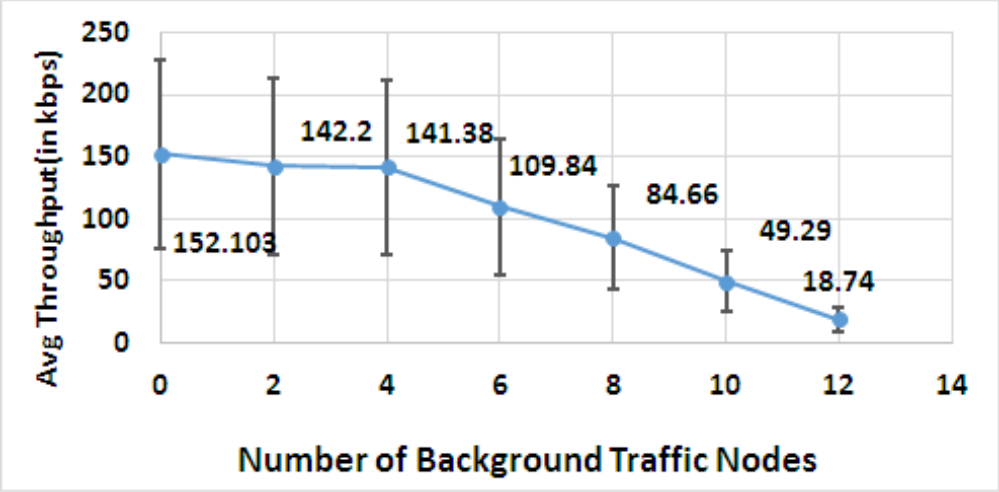

Figure 6.14: Average throughput for WiMAX with HTTP traffic. Reprinted with permission from [6].

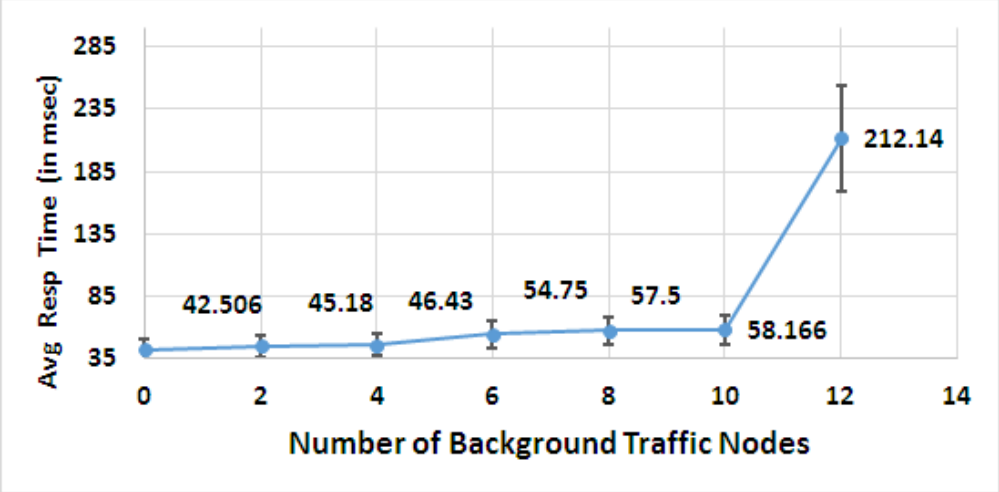

Figure 6.15: Average request/response time for WiMAX using CoAP packets. Reprinted with permission from [6].

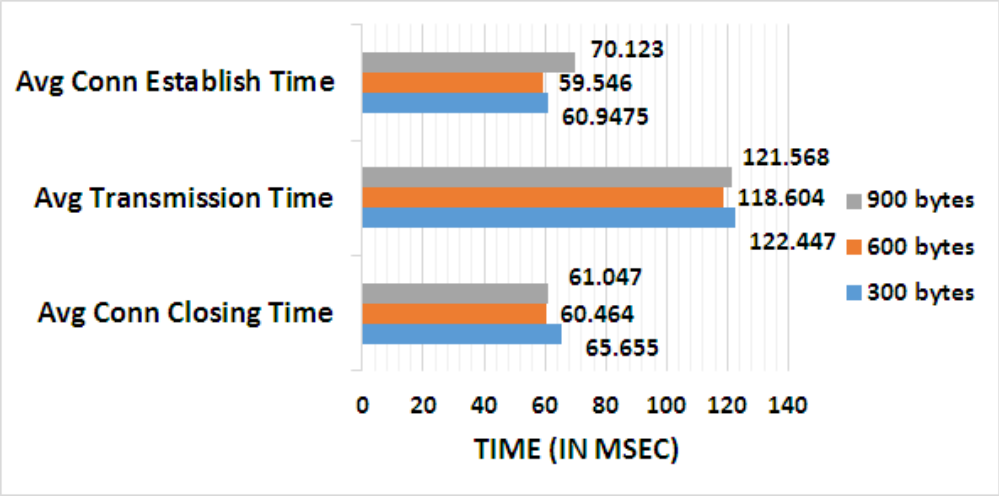

Figure 6.16: LTE HTTP response delay statistics. Reprinted with permission from [6].

6.3.2.3 LTE simulation

In our LTE model, latency and throughput with two smart meters are analyzed. Upon getting the HTTP GET request from the data collection server, the smart meter responds with payloads of different sizes ranging from 300 to 900 bytes. The HTTP server are installed in the smart meters using the NS-3 DCE module as explained in Section 3.3. The mean TCP connection establishment, closing, and transmission times are plotted in Figure 6.16, where we observed that the packet sizes have very little impact on the transmission time because of LTEs' high bandwidth capacity and small number of smart meters. The average throughput of the data increases as the payload size of the traffic increases, as shown in Figure 6.17. Although small payload size makes LTE less bandwidth efficient, it can be a good solution when we have many smart meters. In our future work, we will improve the scalability of the LTE model.

6.3.2.4 Cyber attack test

A platform-independent packet generator, Hyenae 0.36, is used to create the SYN flood attack on the data collection center. SYN packets of payload of 500 bytes are sent to the data collection

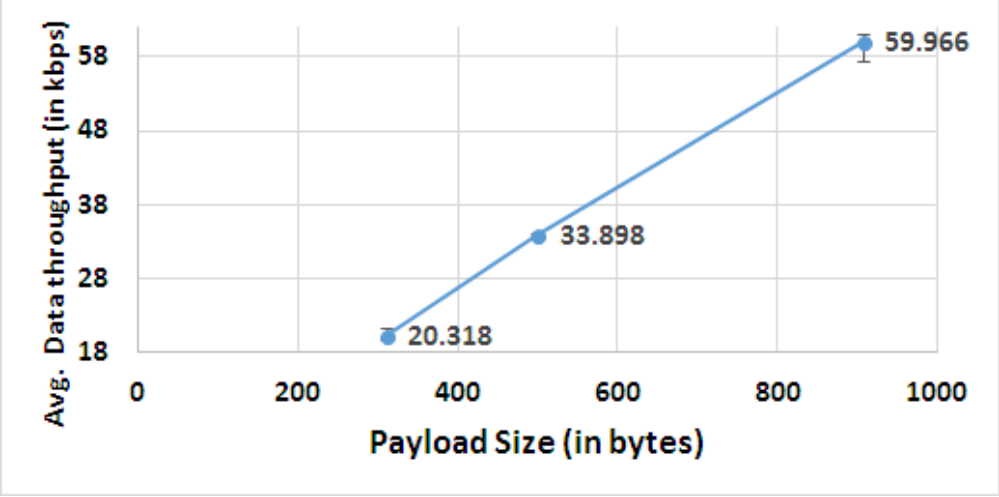

Figure 6.17: Average throughput in LTE for HTTP traffic. Reprinted with permission from [6].

server at intervals ranging from 1 ms to 20 s. The packet capture in Figure 6.18 shows how a WiMAX-based smart meter with IP address 10.10.10.200 communicates with main data collection server with IP address 11.11.11.200. However, after two unsuccessful transmissions, the client sends a CoAP request to the backup data collection server, at IP address 12.12.12.200. The default retransmission timeout for CoAP protocol is 2 s and it doubles with every unsuccessful transmission. We observed that the main data collection server responds to the CoAP requests when the attacking interval is higher than 10 s. For the 10 s attacking interval, the main server responds successfully after one retransmission. For the 20 s attacking interval, the main server always responds to the requests and there are no retransmissions.

6.4 Performance evaluation of C-UNB in NS-3

The performance evaluation of C-UNB based network proposed was evaluated with a lot of assumptions in a python simulator in the Section 6.2. Hence we extended the work to incorporate the detailed specification of C-UNB network in NS-3. The model is designed as described in the Section 4.

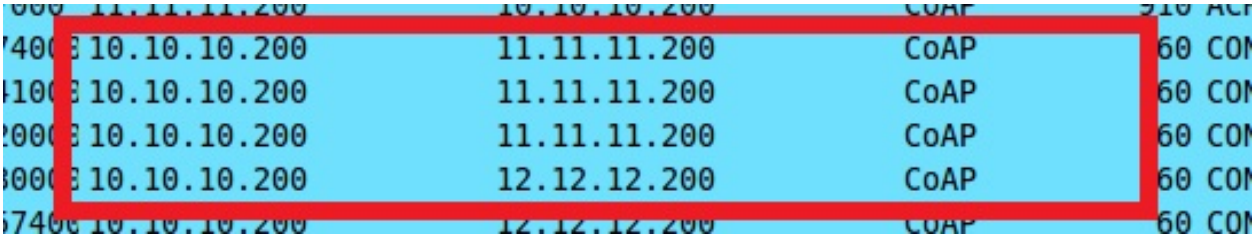

Figure 6.18: Packet capture showing attack on CoAP Server (11.11.11.200), making CoAP client (10.10.10.200) connect with the backup CoAP server (12.12.12.200) after two retransmissions. Reprinted with permission from [6].

6.4.1 Simulation scenario

For simulating C-UNB, the NS-3 simulator need to be configured first. Individual classes of the `cunb` module, illustrated in Section 4, models different components of a C-UNB network. Still, individually each class should be properly integrated with each other for simulation purpose. For that, we use a simulation script that uses the helper classes to configure devices such as smart meters, base stations and C-UNB server. These scripts are also used to collect simulation data through the trace sources that are defined in significant classes: when a specific event occurs during run-time, a script-level function is called to register the event for further analysis.

6.4.1.1 Simulation script

The simulation set up script procedure in the next page, explains the steps taken to set up the C-UNB based AMI network. The packet flow diagram for the simulation is shown in Figure 6.19.

6.4.2 Metrics and parameters for evaluation

The script described requires definition of few parameters and metrics. Some of the parameters are altered to see the effect on some of the metrics for the system. Some featured parameters and metrics are:

a) **Network scale** : Large number of smart meters can create contention resulting in higher probability of two packets colliding. The number of base stations used in the cooperative process will

Procedure Simulation set up script

Input: sm_count = smart meter count,**bs_count** = base station count, **ri** = reporting interval

1. Create the `CunbChannel` object and configure the loss and delay model.
2. Create the smart meters(SM), base stations(eNB) nodes and a C-UNB server.
3. Allocate each SM a random location and eNB a fixed location.
4. Install a `cunb` stack on each SM and eNB node, bind their `CunbPhy` object to the `CunbChannel`.
5. Join the callbacks of the SM and eNB trace sources to the functions in the script.
6. Set up Hello, Mobile Autonomous Reporting and DLMS-COSEM server application at the SMs.
7. Set up DLMS-COSEM client application at the C-UNB server.
8. Start the simulation.
9. Save the simulation results.

effect the coverage of smart meters served in an area. More number of base stations will allow more number of smart meters' data to be successfully received at the C-UNB server.

b) **Traffic generated by the upper layers** : For a given number of base stations, lower reporting interval of the message will cause more collisions between packets.

c) **Retransmission probability**: Retransmission probability mainly depends on the probability of collision of a data or ACK packet, which depends on the receivers' sensitivity.

d) **Channel throughput**: Network efficiency can be computed based on the duration for which the channel is utilized for successful transmission of packets.

6.4.3 Assumptions

1. The effective bandwidth of the channel is considered to be 200 kHz. Within that 200 kHz, excluding the guard band of 20 kHz, we considered 150 micro-channels for all the simulations for the `cunb` model.

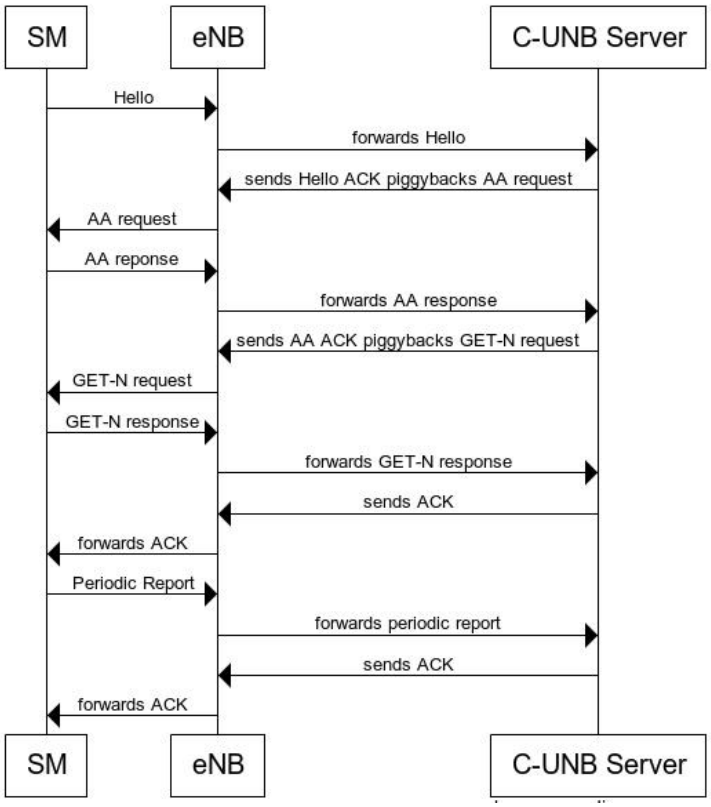

Figure 6.19: Packet flow diagram for DLMS/COSEM and C-UNB based simulation.

2. The Stop and Wait Automatic Repeat Request (ARQ) based retransmission policy is incorporated.

3. Data payload size is kept fixed to be 40 bytes including headers of all the layers.

4. A maximum of three base stations are considered.

5. To analyze the channel throughput we considered lower reporting intervals of 5 to 30 s, as higher reporting intervals cannot utilize the network thoroughly.

6.4.4 Results and analysis

1. The first study on the `cunb` module is conducted to analyze the impact of number of base stations on percentage of retransmissions and successful ACKs. The cooperative logic of C-UNB requires any one of the base station to successfully receive the packet. So for the cases where we use more number of base stations, the percentage of retransmission reduces. As per the Figure 6.20, as the number of base stations increased from 1 to 2, retransmission percentage reduced from 52.5 percent to 26.99 percent for the reporting interval of 2 mins. Similar responses are observed for different Reporting Interval (RI) s. For higher RIs there are very few retransmissions, because with lower RI, the duration for which the channel is occupied reduces which decreases the scope of collision.

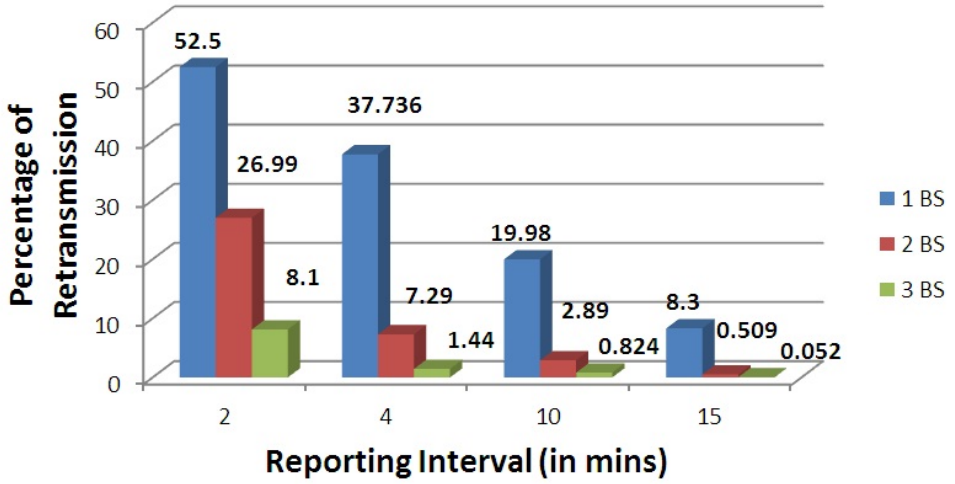

Figure 6.20: Percentage of retransmission for varying reporting interval with different number of base stations.

From Figure 6.21 we can observe that the percentage of successful transmission also increased with the use of more base stations in the simulation. When a smart meter reports data at a RI of

2 mins, 68 percent of packets are successfully acknowledged with one base station, but 89 percent with two base stations followed by 94 percent with three base stations.

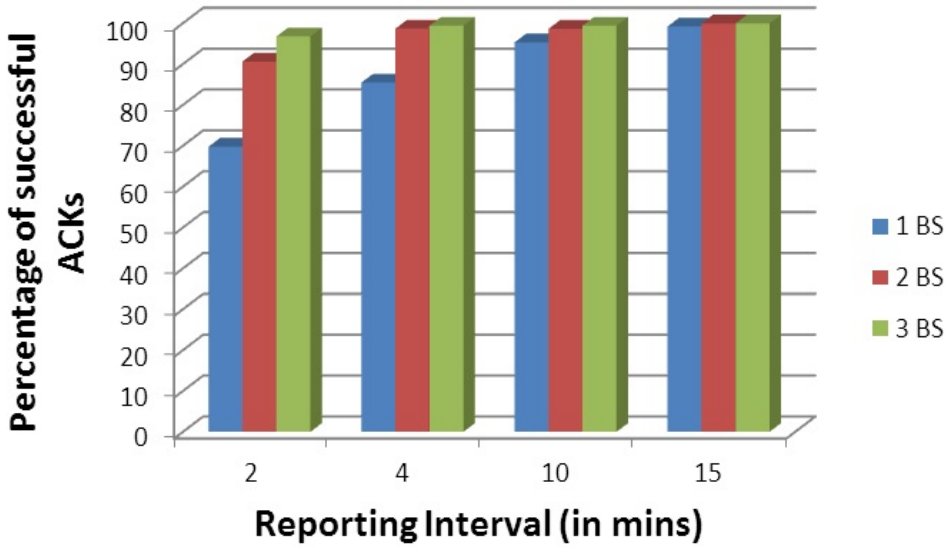

Figure 6.21: Percentage of successful transmissions for varying reporting interval with different number of base stations.

2. For validating the functionalities of the `cunb` module we tried to compare the results from the simulation with the calculated formula on retransmitted probability in Section 2.3. For this simulation we considered two base stations and 40 smart meters each generating data of 40 bytes and receiving ACKs of size 23 bytes for different RIs, since the maximum allowed payload size excluding MAC and link layer header, as per [5] is 32 bytes and 34 bytes for uplink and downlink respectively. As discussed in Section 2.3, the probability of retransmission is given by Equation 6.2.

$$
\begin{aligned}
P(\text{retransmission}) = P(\text{collision between ACKs}) + \\
P(\text{collision between GETREs}) + \\
P(\text{collision between GETRE and ACK}) + \\
P(\text{collision between ACK and GETRE})
\end{aligned}
\tag{6.2}
$$

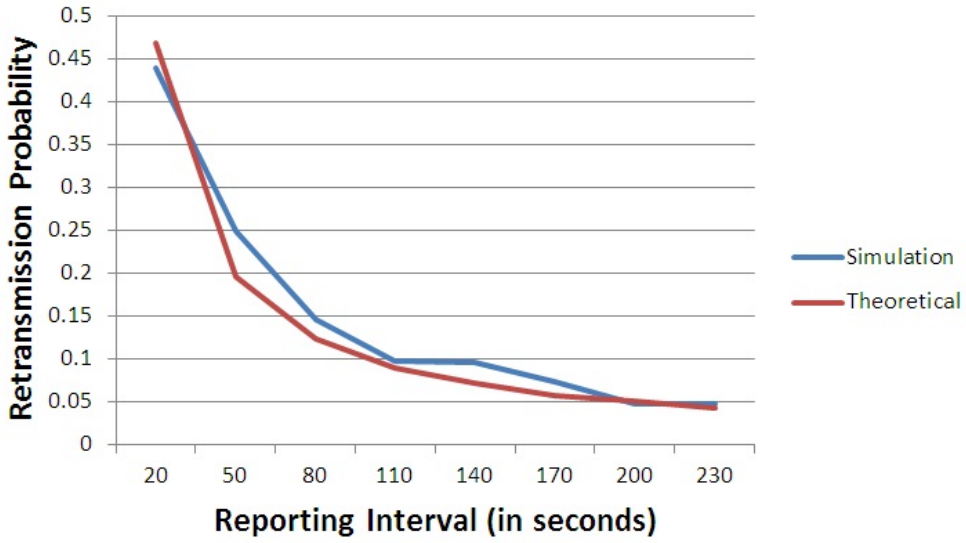

Figure 6.22: Comparison of theoretical calculation and simulation result of retransmission probability with different reporting intervals.

Based on the Equations 2.10, 2.9, 2.7 and 2.8 for all the four expressions in Equation 6.2 respectively, we calculated the retransmission probability to be 0.468 for the RI of 20 s. Figure 6.22 shows the comparison of the calculated value with the simulation results for different RIs. Though the absolute values for the simulated and calculated scenarios are exactly not same but they followed a similar trend.

3. Our next study is performed to observe the impact of number of smart meters on retransmis-

sion and successful transmission. As the number of smart meters increased, the rate of retransmission also increased. From Figure 6.23 we can observe that as the number of smart meters increased from 50 to 250, the retransmission percentage increased from 2.003 percent to 36.93 percent with two base stations and from 0.0097 percent to 16.29 percent with three base stations. And from Figure 6.24 we can conclude that the successful transmission decreases with increase in smart meters. With 50 smart meters almost all the packets are successfully acknowledged.

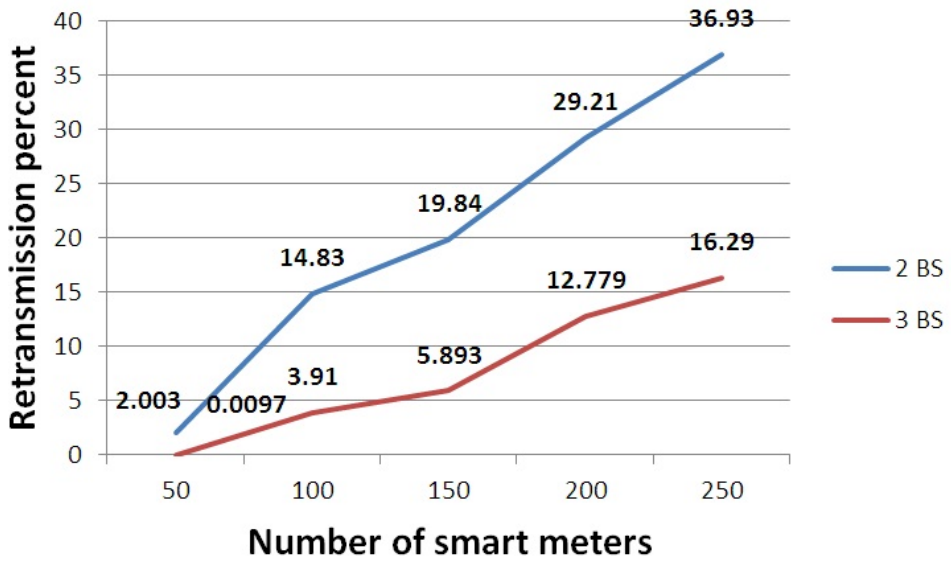

Figure 6.23: Retransmission percent for different number of smart meters.

4. Our final study with regard to the `cunb` module is to evaluate the throughput of the channel, S using Equation 2.13 from Section 2. The network offered traffic, G for all the micro-channels is given by the Equation 2.12. Hence, G per channel is given by Equation 6.3, since 150 micro-channels are considered for the simulation. For this simulation we considered a fixed reporting interval of 5 s and increased the number of smart meters to raise the value of G.

$$G = \sum_{i=1}^{N} (\mathrm{t_{u,i}} + t_{\mathrm{d,i}}) \,/\, (\tau_i * 150) \tag{6.3}$$

It can be observed from the Figure 6.25, that the throughput increases for lower values of G and then it starts to reduce as the network gets overloaded. The throughput of the channel is comparable with the throughput of a Pure-ALOHA protocol except that it provided a maximum throughput of 0.045 at around 40 percent unlike 0.18 at 50 percent of G in Pure-ALOHA.

Hence, the performance evaluation of the novel `cunb` module is successfully executed. Now we will proceed with the simulation for the proposed IDS for ARP spoof detection.

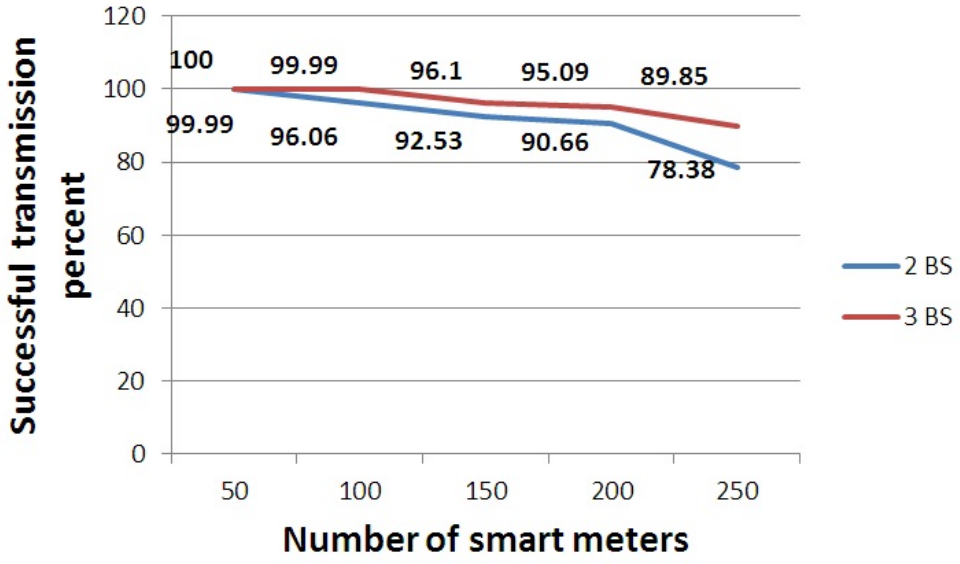

Figure 6.24: Successful transmission percent for different number of smart meters.

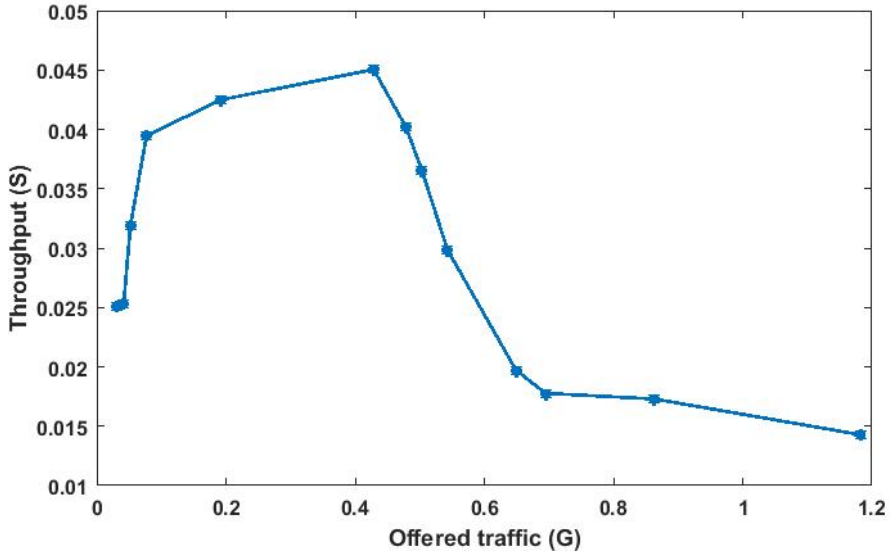

Figure 6.25: Throughput of the C-UNB channel.

6.5 IDS simulation

For testing the IDS we had to add one more computer to our testbed. Our testbed now consists of three computers. One of the computers is the head-end system which is acting as an Hypertext Transfer Protocol (HTTP) client and a Constrained Application Protocol (CoAP) client, where CoAP is a lightweight version of HTTP. Raspberry Pi emulates a smart meter where CoAP and HTTP servers are running. Another laptop is acting as the rogue node (or attack meter) where the Ettercap application is used for ARP poisoning attack. The NS-3 AMI model is running in the third PC.

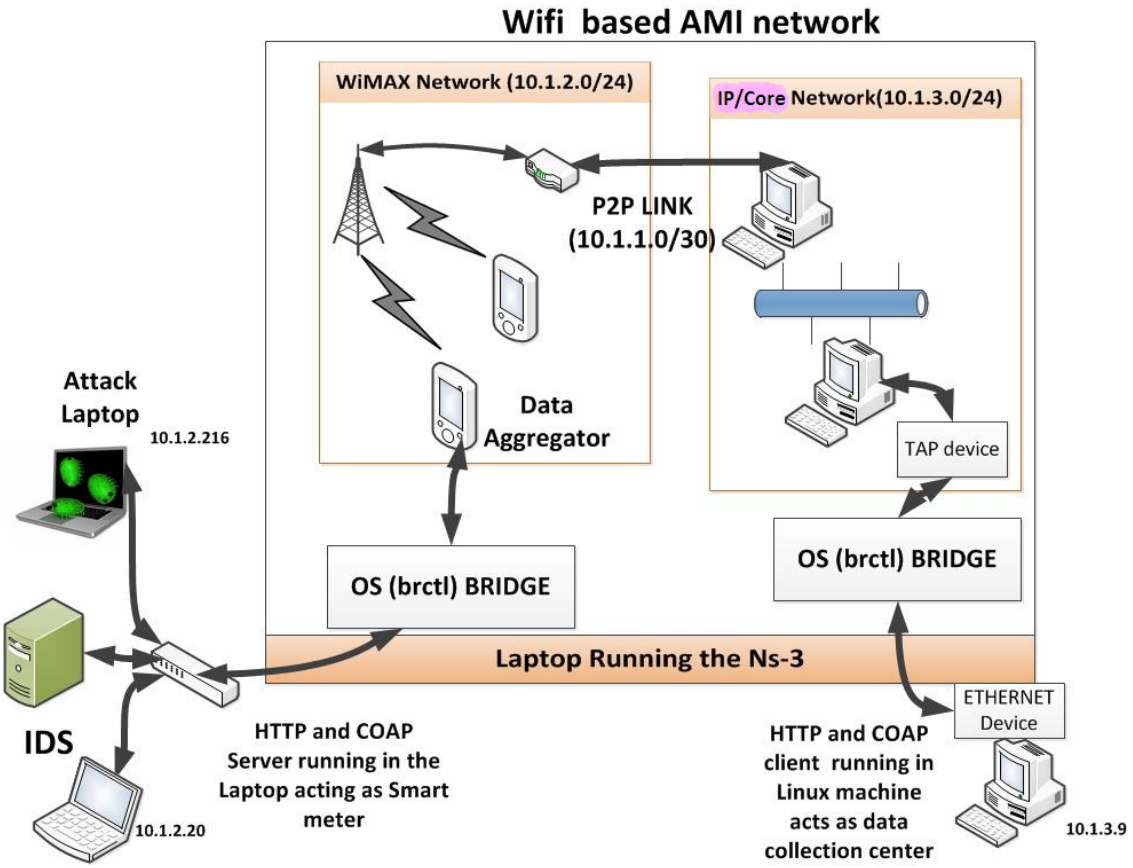

Figure 6.26: NS-3 model for AMI network.

Figure 6.26 shows the AMI network model built using network simulator NS-3. In our model, smart meters and data collection servers see the NS-3 node as a gateway to traverse through the communication channel built inside it.

6.5.1 Results for host-based IDS at smart meter

First, we observe normal communication between the smart meter and head-end system. When we perform the ARP cache poisoning in the AMI network, we observe a rise in the Round Trip Time (RTT). The normal RTT ranged from 5 ms to 7 ms, but after the ARP poisoning it increases to more than 10 ms, as shown in Figure 6.27. We attacked the smart meter six times in a specific time, and we observed the abrupt increase in RTT with some packets resulting in more than 25 ms

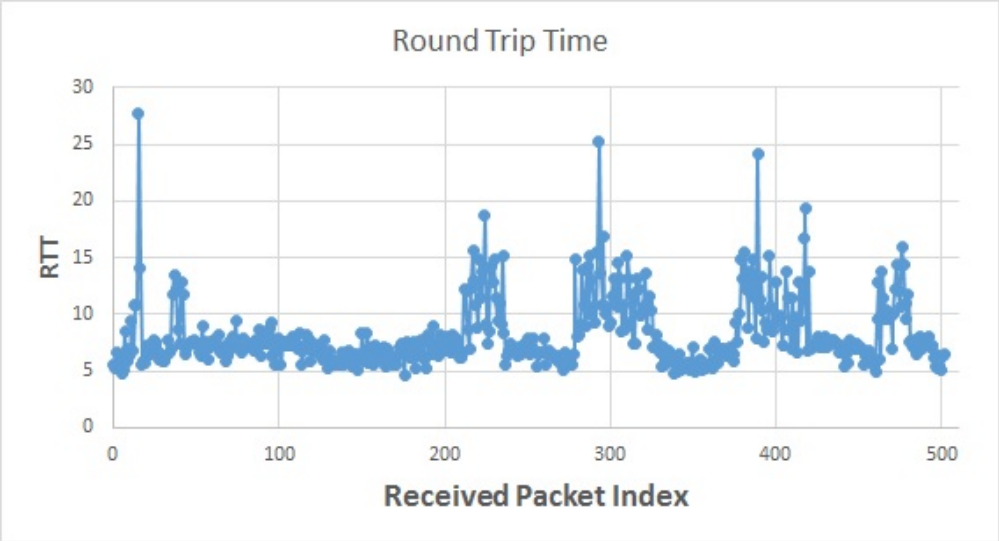

Figure 6.27: Rise in Round Trip Time. Reprinted with permission from [2].

of delay. Along with the rise in RTT, we also observe the occurrence of gratuitous ARP packets (Figure 6.28). The higher frequency of gratuitous ARP marked the sign of an ARP poisoning attack, as we can observe comparing the plot where the RTT has increased and the gratuitous ARP flag has been observed with high frequency. In this simulated case, we can see there are six instances where we observed this scenario.

After observing this behavior, we trained our model to detect the ARP spoofing attack based on these features. The training model follows a Bayesian approach to upgrade the likelihood of an attack. In the attack detection technique, it was observed that some packets were falsely detected as an attack; therefore, we modified our threshold for detection dynamically as mentioned in Equation 5.5. For every 10 packets received by our host-based IDS, it trained our Bayesian model dynamically and concluded whether an attack happened or not. In the experiment, we captured 500 packets and observed 50 instances of the detection logic as shown in Figure 6.29.

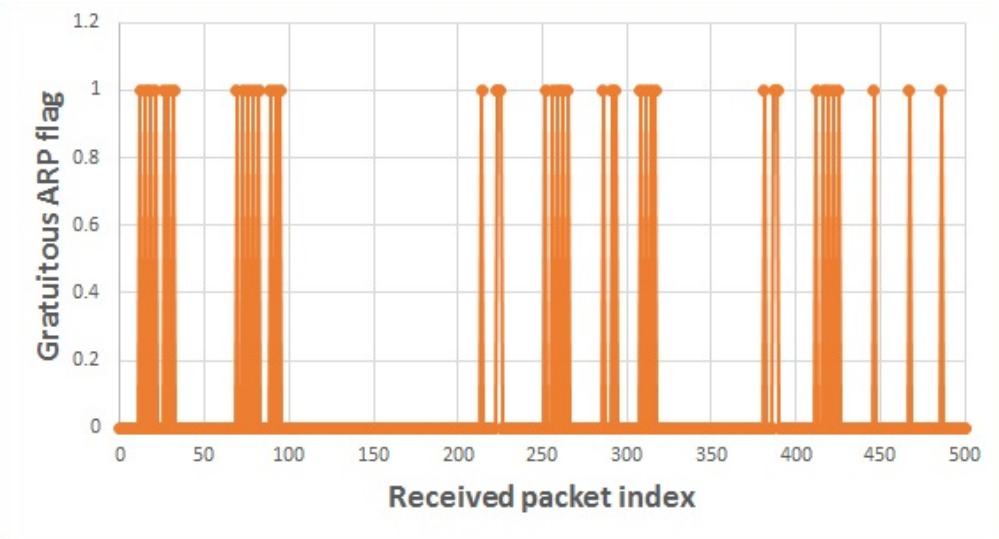

Figure 6.28: Observation of high frequency gratuitous ARP packets. Reprinted with permission from [2].

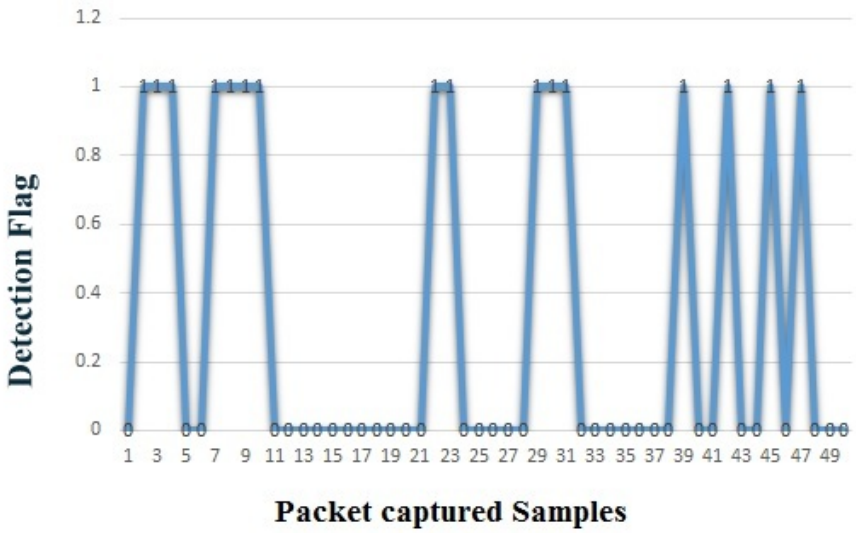

Figure 6.29: Attack detection by the host-based IDS. Reprinted with permission from [2].

6.5.2 Results for network-based IDS at concentrator

The network-based IDS is written in Python using specific libraries, such as `ImpactPacket` and `Pcapy`. To run properly on 64-bit systems, we built these libraries from source code and statically linked them to the `WinPCap` development package libraries. `Pcapy` merely logs packets; `ImpactPacket` does all the work for analysis. `Pcapy` can be used for both sniffing the network and outputting to a file, which can then be read and analyzed using `ImpactPacket`. The dumper object of the `Pcapy` library provides this functionality, and the `open_offline()` function allows static analysis of files. As an example, here is how packets are logged in the network:

```
while(1) :
    (header, packet) = cap.next()
    process_pkt(packet),

# determine whether it's a reply or response and continue
def proc_pkt(pkt):
    dcd = ImpactDecoder.EthDecoder()
    ether = dcd.decode(data)
    if isinstance(ether.child(), impacket.ImpactPacket.ARP):
        arph = ether.child()
        opcode = arph.get_ar_op()
        if opcode == 2:
            proc_response(child)
        elif opcode == 1:
            proc_request(child)
```

```
WARNING: 12/10/2016 11:06:33 PM ARP replies detected from MAC 00:00:00:00:00:03. Request count 2
WARNING: 12/10/2016 11:06:37 PM ARP replies detected from MAC 00:24:7e:6b:b5:d9. Request count 9
WARNING: 12/10/2016 11:06:37 PM ARP replies detected from MAC 00:24:7e:6b:b5:d9. Request count 10
WARNING: 12/10/2016 11:06:46 PM ARP replies detected from MAC 08:00:27:ff:8c:51. Request count 5
WARNING: 12/10/2016 11:06:47 PM ARP replies detected from MAC 08:00:27:ff:8c:51. Request count 6
WARNING: 12/10/2016 11:06:48 PM ARP replies detected from MAC 00:24:7e:6b:b5:d9. Request count 11
ERROR: 12/10/2016 11:06:48 PM ARP Spoofing Detected from MAC Address 00:24:7e:6b:b5:d9
WARNING: 12/10/2016 11:06:48 PM ARP replies detected from MAC 00:24:7e:6b:b5:d9. Request count 12
WARNING: 12/10/2016 11:06:48 PM ARP replies detected from MAC 08:00:27:ff:8c:51. Request count 7
WARNING: 12/10/2016 11:06:49 PM ARP replies detected from MAC 08:00:27:ff:8c:51. Request count 8
WARNING: 12/10/2016 11:06:50 PM ARP replies detected from MAC 08:00:27:ff:8c:51. Request count 9
WARNING: 12/10/2016 11:06:55 PM ARP replies detected from MAC 00:00:00:00:00:03. Request count 3
WARNING: 12/10/2016 11:06:56 PM ARP replies detected from MAC 00:00:00:00:00:03. Request count 4
WARNING: 12/10/2016 11:06:57 PM ARP replies detected from MAC 00:00:00:00:00:03. Request count 5
WARNING: 12/10/2016 11:06:58 PM ARP replies detected from MAC 00:24:7e:6b:b5:d9. Request count 13
WARNING: 12/10/2016 11:06:58 PM ARP replies detected from MAC 00:24:7e:6b:b5:d9. Request count 14
```

Figure 6.30: Detection of ARP spoof in the network-based IDS in the data aggregator. Reprinted with permission from [2].

Once packets are sent into the `reply()` or `response()` processors, the formerly mentioned logic took place to analyze the packets for malicious code. We simultaneously created the log of all the ARP packets flowing through the data collector, as shown in Figure 6.30. Our program is able to successfully detect malicious ARP packets coming through the network and log them as malicious when all conditions were met. Detected rogue nodes are stored in a log file with information about the offending MAC address, all its associated victim IP addresses, the number of gratuitous requests it sent, and the time stamp it was logged. While this is a proof of concept for a larger IDS, it is a valuable example of the simplicity required for an IDS on a data collection center.

Whenever we observe the ARP spoof packets in the host-based or network-based IDS, we automatically run IPTables (i.e., a firewall) to block all the packets originating from the attackers' IP address.

7. CONCLUSION

7.1 Conclusion

In this thesis, we explored different network simulators and wireless technologies to model a real-time simulation testbed for AMI for the evaluation of performance such as throughput, latency, and security. We performed preliminary simulations on a novel C-IoT based, C-UNB technology for the NAN of AMI. Based on the successful evaluation of the model, we developed the C-UNB MAC and physical layer models in NS-3. Besides, we integrated the DLMS-COSEM module of NS-3 into our `cunb` module. We performed a few security study by creating attacks such as DoS and ARP spoofing in our testbed. Finally, we proposed a new Bayesian-based IDS for ARP spoof detection. This work helped me learn the overall aspects of designing a testbed, ranging from modeling a wireless technology to developing an IDS solution.

7.2 Scope of future work

The testbed only models a few wireless technologies for the NAN of AMI network and designs a few attack scenarios. It can be further extended by integrating other cyber-physical attacks like session hijacking, repetition attack, DNS spoofing etc. Further, we can develop a Markov Decision Process based assessment framework like [56] for AMI network to evaluate the resiliency of the network to cyber attacks. The current `cunb` module for NS-3 does not facilitate segmentation and reassembly of large packets, which can be incorporated in future versions. Besides, in the latest `cunb` module, there is no support for the TAP interfaces as we have for Wi-Fi, WiMAX and LTE in NS-3. Hence, the `cunb` module can be improved to support packet capture and TAP bridge features to inject real-time traffic into the C-UNB network.

REFERENCES

[1] "IEEE guide for smart grid interoperability of energy technology and information technology operation with the electric power system (EPS), end-use applications, and loads," *IEEE Std 2030-2011*, pp. 1–126, Sept 2011.

[2] A. Sahu, H. N. R. K. Tippanaboyana, L. Hefton, and A. Goulart, "Detection of rogue nodes in AMI networks," pp. 1–6, Sept 2017.

[3] M. Koseoglu, "Performance analysis of small data transmission schemes for cellular M2M communications," *2017 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pp. 1–6, June 2017.

[4] A. E. Goulart and A. Sahu, "Cellular IoT for mobile autonomous reporting in the smart grid," *Int. J. Interdiscip. Telecommun. Netw.*, vol. 8, pp. 50–65, July 2016.

[5] 3GPP, "Cellular system support for ultra-low complexity and low throughput internet of things (CIoT) (release 13), 3GPP TR 45.820 V13.1.0," *3GPP - Technical Specification Group GSN/EDGE Radio Access Network*, November 2015.

[6] A. Sahu, A. E. Goulart, and K. Butler-Purry, "Modeling AMI network for real-time simulation in NS-3," in *2016 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, pp. 1–8, Oct 2016.

[7] W. Wang and Z. Lu, "Cyber security in the smart grid: Survey and challenges," *Computer Networks*, vol. 57, no. 5, pp. 1344 – 1371, 2013.

[8] R. R. Mohassel, A. Fung, F. Mohammadi, and K. Raahemifar, "A survey on advanced metering infrastructure," *International Journal of Electrical Power Energy Systems*, vol. 63, pp. 473 – 484, 2014.

[9] S. Galli, A. Scaglione, and K. Dostert, "Broadband is power: internet access through the power line network," *IEEE Communications Magazine*, vol. 41, pp. 82–83, May 2003.

[10] M. Kuzlu, M. Pipattanasomporn, and S. Rahman, "Communication network requirements for major smart grid applications in HAN, NAN, and WAN," *Computer Networks*, vol. 67, pp. 74 – 88, 2014.

[11] T. Khalifa, K. Naik, and A. Nayak, "A survey of communication protocols for automatic meter reading applications," *IEEE Communications Surveys Tutorials*, vol. 13, pp. 168–182, Second 2011.

[12] L. Tian, J. Famaey, and S. Latre, "Evaluation of the IEEE 802.11ah restricted access window mechanism for dense IoT networks," pp. 1–9, June 2016.

[13] A. Hazmi, J. Rinne, and M. Valkama, "Feasibility study of 802.11ah radio technology for IoT and M2M use cases," pp. 1687–1692, Dec 2012.

[14] B. Bellekens, L. Tian, P. Boer, M. Weyn, and J. Famaey, "Outdoor IEEE 802.11ah range characterization using validated propagation models," pp. 1–6, Dec 2017.

[15] T. P. C. d. Andrade, C. A. Astudillo, L. R. Sekijima, and N. L. S. d. Fonseca, "The random access procedure in long term evolution networks for the internet of things," *IEEE Communications Magazine*, vol. 55, pp. 124–131, March 2017.

[16] J. J. Nielsen, G. C. Madueno, N. K. Pratas, R. B. Sorensen, C. Stefanovic, and P. Popovski, "What can wireless cellular technologies do about the upcoming smart metering traffic?," *IEEE Communications Magazine*, vol. 53, pp. 41–47, September 2015.

[17] S. G. Kerk, "An AMR study in an indian utility," pp. 1–142, Nov 2005.

[18] S. C. Soh and S. G. Kerk, "The electricity and metering trends in singapore," pp. 1–152, Nov 2005.

[19] A. Sahu and S. Behera, "PAPR analysis and channel estimation techniques for 3GPP LTE system," *http://ethesis.nitrkl.ac.in/2648/*, 2011.

[20] E. Weingartner, H. vom Lehn, and K. Wehrle, "A performance comparison of recent network simulators," pp. 1–5, June 2009.

[21] NS-3 Project, "NS-3 sockets API," *https://www.nsnam.org/docs/models/html/sockets-api.html*, 2010.

[22] G. F. Riley, "Large-scale network simulations with GTNetS," vol. 1, pp. 676–684 Vol.1, Dec 2003.

[23] L. Begg, W. Liu, K. Pawlikowski, S. Perera, and H. Sirisena, "Survey of simulators of next generation networks for studying service availability and resilience," *Technical Report TR-COSC 05/06, Department of Computer Science Software Engineering, University of Canterbury, Christchurch, New Zealand*, February, 2006.

[24] S. Duflos, G. L. Grand, A. A. Diallo, C. Chaudet, A. Hecker, C. Balducelli, F. Flentge, C. Schwaegerl, and O. Seifert, "Deliverable d 1.3.2: List of available and suitable simulation components," *Technical report, Ecole Nationale Supieure des Tommunications (ENST)*, September,2006.

[25] K. D. Craemer and G. Deconinck, "Analysis of state-of-the-art smart metering communication standards," *https://lirias.kuleuven.be/bitstream/123456789/265822/1/SmartMeteringCommStandards.pdf*.

[26] P. Fuchs and T. Schaub, "DLMS user association-co-ordination between applications and channels [automatic meter reading]," pp. 124–128, Aug 1999.

[27] S. Feuerhahn, M. Zillgith, C. Wittwer, and C. Wietfeld, "Comparison of the communication protocols DLMS/COSEM, SML, and IEC 61850 for smart metering applications," pp. 410–415, Oct 2011.

[28] Derr and Kurt, "NS-3 web-based user interface: Power grid communications planning and modeling tool," pp. 93–100, 2016.

[29] S. Rana, H. Zhu, C. W. Lee, D. M. Nicol, and I. Shin, "The not-so-smart grid: Preliminary work on identifying vulnerabilities in ANSI C12.22," pp. 1514–1519, Dec 2012.

[30] M. V. Tripunitara and P. Dutta, "A middleware approach to asynchronous and backward compatible detection and prevention of ARP cache poisoning," pp. 303–309, 1999.

[31] P. McDaniel, W. Enck, and W. Lootah, "TARP: Ticket-based address resolution protocol," vol. 00, pp. 106–116, 12 2005.

[32] B. Issac and L. A. Mohammed, "Secure unicast address resolution protocol (S-UARP) by extending DHCP," vol. 1, pp. 6 pp.–, Nov 2005.

[33] H. Ma, H. Ding, Y. Yang, Z. Mi, J. Y. Yang, and Z. Xiong, "Bayes-based ARP attack detection algorithm for cloud centers," *Tsinghua Science and Technology*, vol. 21, pp. 17–28, Feb 2016.

[34] K. A. Ahmed, Z. Aung, and D. Svetinovic, "Smart grid wireless network security requirements analysis," pp. 871–878, Aug 2013.

[35] R. Berthier, D. I. Urbina, A. A. Cardenas, M. Guerrero, U. Herberg, J. G. Jetcheva, D. Mashima, J. H. Huh, and R. B. Bobba, "On the practicality of detecting anomalies with encrypted traffic in AMI," pp. 890–895, Nov 2014.

[36] T. Norp and B. Landais, "M2M communications: A systems approach, M2M optimization for public mobile networks (chapter 6)," *John Wiley Sons*, 2012.

[37] P. Osti, P. Lassila, S. Aalto, A. Larmo, and T. Tirronen, "Analysis of PDCCH performance for M2M traffic in LTE," *IEEE Transactions on Vehicular Technology*, vol. 63, pp. 4357–4371, Nov 2014.

[38] D. Bertsekas and R. Gallager, "Data networks," *Prentice Hall*, vol. Second Edition, 1992.

[39] N. Benvenuto and M. Zorzi, "Principles of communications networks and systems," *Wiley Online Library*, 2011.

[40] NS-3 Project, "NS-3 real-time documentation," *https://www.nsnam.org/docs/manual/html/realtime.html*, 2010.

[41] C. B. Vellaithurai, "Cyber power system analysis using a real-time testbed," *Master Thesis, Washington State University*, 2013.

[42] "TUN-TAP linux interface," *http://backreference.org/2010/03/26/tuntap-interface-tutorial/*, 2010.

[43] NS-3 Project, "NS-3 direct code execution," *https://www.nsnam.org/overview/projects/direct-code-execution/*, 2010.

[44] NS-3 Project, "NS-3 Wi-Fi design documentation," *https://www.nsnam.org/docs/models/html/wifi-design.html*, 2010.

[45] S. Andreev, O. Galinina, A. Pyattaev, M. Gerasimenko, T. Tirronen, J. Torsner, J. Sachs, M. Dohler, and Y. Koucheryavy, "Understanding the IoT connectivity landscape: a contemporary M2M radio technology roadmap," *IEEE Communications Magazine*, vol. 53, pp. 32–40, September 2015.

[46] NS-3 Project, "NS-3 WiMAX design documentation," *https://www.nsnam.org/docs/release/3.13/models/html/wimax.html*, 2010.

[47] H. K. Sharma, S. Sahu, and S. Sharma, "Article: Enhanced cost231 w.i. propagation model in wireless network," *International Journal of Computer Applications*, vol. 19, pp. 36–42, April 2011. Full text available.

[48] NS-3 Project, "NS-3 LTE design documentation," *https://www.nsnam.org/docs/models/html/lte-design.html*, 2010.

[49] NS-3 Project, "NS-3 packet tag," *https://www.nsnam.org/docs/release/3.11/models/html/packets.html*, 2010.

[50] L. Peterson and B. Davie, "Computer networks - a systems approach (chapter 6)," *Morgan Kaufman, Fifth Edition*, march 2011.

[51] Z. C. Dong, R. Espejo, Y. Wan, and W. Zhuang, "Detecting and locating man-in-the-middle attacks in fixed wireless networks.," *CIT*, vol. 23, no. 4, pp. 283–293, 2015.

[52] Grotto Networking, "Basic network simulations and beyond in python," *https://www.grotto-networking.com/DiscreteEventPython.html*, December,2015.

[53] L. Costantino, N. Buonaccorsi, C. Cicconetti, and R. Mambrini, "Performance analysis of an LTE gateway for the IoT," *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–6, June 2012.

[54] O. Bergmenn, "LIBCOAP: C implementation of CoAP," *https://libcoap.net/*, 2015.

[55] J. Poskanzer, "THTTP web server," *http://acme.com/software/thttpd/*, 2014.

[56] K. R. Davis, C. M. Davis, S. A. Zonouz, R. B. Bobba, R. Berthier, L. Garcia, and P. W. Sauer, "A cyber-physical modeling and assessment framework for power grid infrastructures," *IEEE Transactions on Smart Grid*, vol. 6, pp. 2464–2475, Sept 2015.

APPENDIX A

LIST OF IMPORTANT CLASSES AND FUNCTIONS OF THE NS-3 `CUNB` MODULE

This section comprises of the documentation for the C-UNB module created in NS-3.

Table A.1: C-UNB Physical Layer (`CunbPhy`)

Sl. No.	Function	Description
1.	`StartReceive()`	Start receiving a packet. This method is called by `CunbChannel`, after a delay of the `Send()` function call in the `CunbChannel` based on the `PropagationDelayModel`.
2.	`EndReceive()`	Finish reception of a packet. This method is scheduled by `StartReceive`, based on the packet transmission time. By passing a `CunbInterferenceHelper` Event to this method, the class receives the packet that is being received among all those that were not registered as interference by `StartReceive`.
3.	`Send()`	Instructs the physical layer to send a packet according to parameters like frequency, transmit power etc.
4.	`IsTransmitting()`	Checks whether this device is transmitting or not.
5.	`IsOnFrequency()`	Checks whether this device is listening on the specified frequency.

Table A.1 Continued..

Sl. No.	Function	Description
6.	`SetRcvOkCallback()`	Set the callback to call upon successful reception of a packet.
7.	`SetTxFinCallback()`	Set the callback to call after transmission of a packet.
8.	`GetMobility()`	Get the mobility model associated to the `CunbPhy`.
9.	`SetMobility()`	Set the mobility model associated to `CunbPhy`.
10.	`SetChannel()`	Set the `CunbChannel` instance `CunbPhy` transmits on.
11.	`GetChannel()`	Get the `CunbChannel` instance associated to the `CunbPhy`.
12.	`GetDevice()`	Get the `NetDevice` associated to the `CunbPhy`.
13.	`SetDevice()`	Set the `NetDevice` associated to the `CunbPhy`.
14.	`GetOnAirTime()`	Compute the time a packet takes to be transmitted. Besides from the ones saved in `CunbTxParameters`, the packets' payload(obtained through a `GetSize ()` call to account for the presence of headers and trailers, too) also influences the packet transmit time.
14.	`SetRxReqOkCallback()`	Set the callback upon successful reception of a request.

Table A.2: C-UNB Channel (`CunbChannel`)

Sl. No.	Function	Description
1.	`Add()`	Connects a `CunbPhy` object to the `CunbChannel`. This method is needed so that the channel knows it has to notify this physical object of incoming transmissions.
2.	`Remove()`	Removes a `CunbPhy` object from the list of devices from the `CunbChannel`. Removing unused physical layers from the channel improves performance.
3.	`Send()`	Invoked by a `CunbPhy` object that needs to send a packet. Every connected physical object will be notified of this packet send through a call to their `StartReceive` methods after a delay based on the channels' `PropagationDelayModel`.
4.	`GetRxPower()`	Computes the received power when transmitting from one node to another. This method can be used by external object to check the received power of a transmission from one point to another using this channel's `PropagationLossModel`.
5.	`Receive()`	Private method that is scheduled by `CunbChannels`' `Send` method to happen after the channel delay, for each of the connected `CunbPhy` object. It's here that the `Receive()` method of the physical layer is called to initiate packet reception at the PHY.

Table A.3: C-UNB Base Station Physical (`EnbCunbPhy`)

Sl. No.	Function	Description
1.	`AddReceptionPath()`	Add a reception path, locked on a specific frequency.
2.	`ResetReceptionPaths()`	Reset the list of reception paths. This method deletes all currently available `ReceptionPath` objects.

Table A.4: Reception Path (`ReceptionPath`)

Sl. No.	Function	Description
1.	`GetFrequency()`	Gets the operating frequency.
2.	`SetFrequency()`	Sets the frequency.
3.	`IsAvailable()`	Query whether this reception path is available to lock on a signal.
4.	`Free()`	Set this reception path as available. This function sets the `m_available` variable as true, and deletes the `CunbInterferenceHelper` Event this `ReceivePath` was previously locked on.
5.	`LockOnEvent()`	Set this reception path as not available and lock it on the provided event.
6.	`SetEvent()`	Set the event this reception path is currently on.
7.	`GetEvent()`	Get the event this reception path is currently on.

Table A.5: C-UNB Smart Meter Physical (`MSCunbPhy`)

Sl. No.	Function	Description
1.	`SetFrequency()`	Set the frequency the `MS` will listen on. If a packet is transmitted on a different frequency than that the `MSCunbPhy` is listening on, the packet is discarded.
2.	`GetState()`	Return the state this `MS` is currently in.
3.	`SwitchToStandby()`	Switch to STANDBY state
4.	`SwitchToRx()`	Switch to RX state
5.	`SwitchToTX()`	Switch to TX state

Table A.6: C-UNB MAC layer (`CunbMac`)

Sl. No.	Function	Description
1.	`SetPhy()`	Set the underlying `CunbPhy`
2.	`GetPhy()`	Get the underlying `CunbPhy`
3.	`Send()`	Send a packet to `CunbPhy`
4.	`Receive()`	Receive a packet from the lower layer
5.	`TxFinished()`	Perform actions after sending a packet.
6.	`SetDevice()`	Set the device this MAC layer is installed on.
7.	`GetDevice()`	Get the device this MAC layer is installed on.
8.	`GetDbmForTxPower()`	Get the transmission power in dBm.
9.	`SetTxDbmForTxPower()`	Set the transmission power in dBm.

Table A.7: Smart Meter MAC layer (`MSCunbMac`)

Sl. No.	Function	Description
1.	`Send()`	Send a packet. The MAC layer of the MS will take care of using the right parameters.
2.	`Receive()`	Receive a packet. This method is typically registered as a callback in the underlying physical layer so that its' called when a packet is going up the stack.
3.	`TxFinished()`	Perform the actions that are required after a packet sends.
4.	`OpenFirstRxWindow()`	Perform operations needed to open the first receive window.
5.	`OpenSecondRxWindow()`	Perform operations needed to open the second receive window.
6.	`OpenThirdRxWindow()`	Perform operations needed to open the third receive window.
7.	`CloseFirstRxWindow()`	Perform operations needed to close the first receive window.
8.	`CloseSecondRxWindow()`	Perform operations needed to close the second receive window.
9.	`CloseThirdRxWindow()`	Perform operations needed to close the third receive window.
10.	`SetDataRate()`	Set the data rate this `MS` will use when transmitting. For MS this value is assumed to be fixed.

Table A.7 Continued..

Sl. No.	Function	Description
11.	`GetDataRate()`	Get the data rate this end device is set to use.
12.	`SetDeviceAddress()`	Set the network address of this device.
13.	`GetDeviceAddress()`	Get the network address of this device.
14.	`SetMType()`	Set the message type to send when the `Send` method is called.
15.	`AddLogicalChannel()`	Add a logical channel to the helper.
16.	`SetLogicalChannel()`	Set a new logical channel in the helper.
17.	`GetChannelForTx()`	Find a suitable channel for transmission. The channel is chosen among the ones that are available in the `MSs' LogicalCunbChannel`.

Table A.8: C-UNB Base Station MAC (`EnbCunbMac`)

Sl. No.	Function	Description
1.	`GetWaitingTime()`	Return the next time at which it will transmit.

Table A.9: C-UNB MAC header (`CunbMacHeader`)

Sl. No.	Function	Description
1.	`Serialize()`	Serialize the header.
2.	`Deserialize()`	Deserialize the header.

Table A.10: C-UNB MAC trailer (`CunbMacTrailer`)

Sl. No.	Function	Description
1.	`Serialize()`	Serialize the header.
2.	`Deserialize()`	Deserialize the header.
3.	`SetAuth()`	Sets the authentication header with the hash returned from `GenerateHash()`.
4.	`CheckAuth()`	Verifies the authentication header by using the secret key, sequence number, identifier and payload.
5.	`SetFcs()`	Set the FCS field with the CRC-8 returned by the `GenerateCrc8()`.
6.	`CheckFcs()`	Verifies the fcs header by using the payload.
7.	`GenerateHash()`	Generate the hash using the SHA-1 algorithm and returns the first 16 bits from the 128 bits hash.
8.	`GenerateCrc8()`	Generate CRC using the payload.

Table A.11: C-UNB server (`SimpleCunbServer`)

Sl. No.	Function	Description
1.	`AddNodes()`	Informs the `SimpleCunbServer` that these nodes are connected to the network. This method creates a `MSStatus` object for each new node added.
2.	`AddEnb()`	Add the eNB connected to this C-UNB server.
3.	`Receive()`	Receives a packet from an eNB.

Table A.11 Continued..

Sl. No.	Function	Description
4.	`SendOnFirstWindow()`	Send a packet through eNB using first receive window.
5.	`GetEnbForReply()`	Gets the best eNB that is available to reply based on the received power.
6.	`PaiExist()`	Checks if a packet containing the pair of identifier and sequence number exists. This is to deduplicate the packets sent by the `MS`
7.	`RemoveOldPair()`	Remove old pairs of identifier and sequence number from the list once the sequence number reaches the maximum value of 255.
8.	`SendRequest()`	Sends AARE and GETRE requests.

Table A.12: Mobile Autonomous Reporting (`MobileAutonomousReporting`)

Sl. No.	Function	Description
1.	`SetInterval()`	Sets the sending interval.
2.	`GetInterval()`	Gets the sending interval.
3.	`SetInitialDelay()`	Sets the initial offset.
4.	`SendPacket()`	Send a packet using the `CunbNetDevices`' `Send` method.
5.	`StartMAR()`	This method is scheduled in the MS on successful association with the C-UNB server.

Table A.13: MS Status (`MSStatus`)

Sl. No.	Function	Description
1.	`UpdateEnbData()`	Update the `MSStatus` for considering the power with which a packed was received by eNB.
2.	`GetBestEnbAddress()`	Return the address of the eNB that received this mobile stations' last packet with highest power.
3.	`GetSortedEnbAddress()`	Return an iterator to the eNB addresses that received a packet for this mobile station.

Table A.14: eNB Status (`EnbStatus`)

Sl. No.	Function	Description
1.	`IsAvailableForTx()`	Query whether or not the eNB is available for immediate transmission.
2.	`SetNextTransmissionTime()`	Sets the next transmission time if there is no immediate available eNB.
3.	`GetNextTransmissionTime()`	Gets the next transmission time.

Table A.15: C-UNB forwarder (`CunbForwarder`)

Sl. No.	Function	Description
1.	`SetCunbNetDevice()`	Sets the eNB to communicate with the mobile station.
2.	`SetP2PNetDevice()`	Sets the eNB to communicate with the Core network.
3.	`RxFromCunb()`	Receives a packet from the `CunbNetDevice`.

Table A.15 Continued..

Sl. No.	Function	Description
4.	`RxFromP2P()`	Receives packet from `PointToPointNetDevice`.

Table A.16: C-UNB net device (`CunbNetDevice`)

Sl. No.	Function	Description
1.	`SetMac()`	Sets the `CunbMac` linked to this device.
2.	`GetMac()`	Gets the `CunbMac` linked to this device.
3.	`SetPhy()`	Sets the `CunbPhy` linked to this `CunbNetDevice`.
4.	`GetPhy()`	Gets the `CunbPhy` linked to this `CunbNetDevice`.
5.	`Send()`	Send a packet through the C-UNB stack.
6.	`Receive()`	Callback the MAC layer calls whenever a packet arrives and needs to be forwarded to the upward layers.

Table A.17: One Time Reporting (`OneTimeReporting`)

Sl. No.	Function	Description
1.	`SetSendTime()`	Sets the send time for AARE packet for the mobile station.
2.	`ReceiveRequest()`	Receives GETRQ and AARQ request
3.	`SendPacket()`	Sends GETRE and AARE based on the request