Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2015

Tiago Rafael Correia Almeida

Delay Tolerant Network for Navy Scenarios: Quality-based Approach

Rede Tolerante a Atraso para Cenários da Marinha: Abordagem baseada na Qualidade

"The measure of who we are is what we do with what we have."

- Vince Lombardi

Universidade de Aveiro Departamento de
2015 Electrónica, Telecomunicações e Informática,

Tiago Rafael Correia Almeida

Delay Tolerant Network for Navy Scenarios: Quality-based Approach

Rede Tolerante a Atraso para Cenários da Marinha: Abordagem baseada na Qualidade

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica da Professora Doutora Susana Sargento, Professora Associada com Agregação do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e co-orientação científica do Doutor Lucas Guardalben, Investigador do Instituto de Telecomunicações de Aveiro.

o júri / the jury

presidente / president	**Professor Doutor Rui Luís Andrade Aguiar** Professor Catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro
vogais / examiners committee	**Professor Doutor Augusto José Venâncio Neto** Professor Adjunto da Universidade Federal do Ceará (Arguente)
	Professora Doutora Susana Isabel Barreto de Miranda Sargento Professora Associada com Agregação do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (Orientadora)

agradecimentos / aknowledgments

Este trabalho marca o fim do meu percurso académico marcado por inúmeras experiências que contribuíram para o meu desenvolvimento, quer a nível pessoal, quer a nível profissional. Agradeço a todos que de alguma forma contribuíram para este desenvolvimento, com especial destaque para os que se seguem.

Antes de mais, agradeço profundamente aos meus Pais e Avó pela educação que me foi transmitida e por todo o apoio que me tem sido dado durante toda a minha vida. Sem vocês não estaria a atingir esta meta. Muito Obrigado!

Um agradecimento especial aos meus grandes amigos e colegas Marco Oliveira, Gonçalo Pessoa, Bojan Magusic, André Martins, Gonçalo Gomes e Rodrigo Almeida com quem tive o prazer de conviver e partilhar o laboratório de redes durante este último ano. Obrigado por todo o apoio e companheirismo malta!

Agradeço ainda à Diana, Fábio, Adriana, Liliana, Vanessa, Inês, André, Rúben, Rui, Miguel, Deolinda, Diogo e restantes amigos pela amizade e pela compreensão nos momentos em que estive mais ausente ou com menos paciência durante este último ano. Obrigado pessoal!

Agradeço à Professora Doutora Susana Sargento e ao Doutor Lucas Guardalben por me terem sugerido este trabalho e por toda a orientação e apoio fornecidos que o levaram rumo a bom porto. Muito Obrigado pela experiência!

Para terminar, agradeço ao grupo de investigação do NAP pelo apoio, ao Instituto de Telecomunicações de Aveiro pelos recursos facultados e a todos os professores que tive o prazer de conhecer e que contribuíram para o meu sucesso académico. Obrigado a todos!

Palavras-chave

Redes Tolerantes a Atrasos, Routing, Qualidade das Ligações, Disrupções, Marinha, IEEE 802.11, Linux Wireless Subsystem.

Resumo

As operações da marinha envolvem vários intervenientes que trabalham entre si com objetivos comuns e frequentemente sob condições de comunicação desafiadoras. Existem constrangimentos naturais que são impostos pelo ambiente da operação, por exemplo, geografia acidentada do terreno. Existem também constrangimentos artificiais que são criados por elementos hostis que forçam condições de modo a prejudicar as operações da marinha (ou outras equipas militares), por exemplo, criação de interferência intencional. Os militares geralmente usam equipamentos de comunicação proprietários para comunicar entre si. Apesar da eficácia destes equipamentos, eles são caros e normalmente oferecem uma gama de serviços limitada. Contudo, os recentes avanços tecnológicos permitiram a proliferação de muitos dispositivos portáteis com capacidade de comunicação sem fios e com o valor de acrescentar novas funcionalidades de formas muito simples, mas estes dispositivos ainda não estão adaptados para as redes militares em termos de comunicação. Esta dissertação propõe usar Redes Tolerantes a Atrasos (DTNs) com um novo protocolo de encaminhamento Quality-PRoPHET (Q-PRoPHET) capaz de medir a qualidade das ligações sem-fios e encaminhar a informação pelas ligações de melhor qualidade, onde a probabilidade de sucesso da transmissão é maior. O Q-PRoPHET usa uma função de qualidade para avaliar a qualidade das ligações e uma propriedade transitiva para encaminhamento a múltiplos saltos. Este algoritmo foi implementado no IBR-DTN e foi avaliado em três cenários que emulam três cenários observados durante operações táticas da Marinha. Dois destes cenários foram testados dentro de um edifício e o último foi testado em ambiente exterior, recorrendo a mobilidade real dos nós. Os resultados obtidos mostram que o Q-PRoPHET tem melhor desempenho que o PRoPHET em termos de taxa de entrega, tempo de entrega e transmissão de pacotes, que são parâmetros críticos para as comunicações das operações da marinha.

Keywords Delay Tolerant Networks, Routing, Link Quality, Disruptions, Navy, IEEE 802.11, Linux Wireless Subsystem.

Abstract

The navy operations involve several participants that work between them with common objectives and usually under challenged communication conditions. There are natural constrains that are imposed by the operation environment, e.g. hilly terrains. There are also artificial constrains that are created by enemy elements which force conditions to affect the navy operation (or other military forces), e.g. intentional jamming. The military often uses proprietary devices to communicate between them. Despite of the effectiveness of these devices, they are expensive and usually offer a limited range of services. However, the recent technological advances allow the proliferation of several mobile devices with wireless communication capabilities and with the value to easily insert new features, but these devices are still not prepared to military networks in terms of communication. Thus, this dissertation proposes to use Delay Tolerant Networks (DTNs) with a new routing protocol Quality-PRoPHET (Q-PRoPHET) able to measure the quality of the wireless links and route the information using the connections with best quality, where the probability of transmission is higher. The Q-PRoPHET uses a quality function to evaluate the quality of the connections and a transitive property to route through multiple hops. This algorithm was implemented in IBR-DTN and it was evaluated in three scenarios that emulate three scenarios observed during the navy tactical operations. Two of these scenarios were tested inside a building and the last one was tested in an external environment using real mobility of the nodes. The obtained results show that Q-PRoPHET has better performance than PRoPHET in terms of delivery ratio, end-to-end delay and packets transmission, which are critical parameters for the communication in navy operations.

Contents

List of Figures

List of Tables

List of Equations

List of Algorithms

Acronyms

ADU	Application Data Unit
AP	Access Point
API	Application Programming Interface
BPF	Berkeley Packet Filtering
BSP	Bundle Security Protocol
BSS	Basic Service Set
CECOM	Army Communications-Electronics Command
CLA	Convergence Layer Adapter
CONDOR	C2 On-the-move Network Digital Over-the-horizon Relay
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
dB	decibel
dBm	decibel miliwatt
DCF	Distributed coordination function
DHCP	Dynamic Host Configuration Protocol
DiPRoPHET	Distance-based PRoPHET
DS	Domain System
DTLSR	Delay Tolerant Link State Routing
DTN	Delay Tolerant Network
e2e	end-to-end

EID	Endpoint Identifier
ESS	Extended Service Set
ETSI	European Telecommunications Standards Institute
FB	Foreign Boat
FCS	Frame Check Sequence
GIG	Global Information Grid
HCF	Hybrid Coordination Function
HQ	Headquarters
HT	High Throughput
HTTP	Hypertext Transfer Protocol
HTTPCL	HTTP Convergence Layer
IB	Intermediate Boat
IBSS	Independent Basic Service Set
ID	Identification
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPC	Inter Process Communication
IPN	Interplanetary Internet
IPND	DTN IP Neighbor Discovery
IPNRG	Interplanetary Internet Research Group
ISA	Instruction Set Architecture
LAN	Local Area Network
LowPANCL	LowPAN Convergence Layer
LStab	Link Stability
LTP	Licklider Transmission Protocol

MAC	Media Access Control
MANET	Mobile Ad-Hoc Network
MGR	Minimum Reception Group
MS	Mother Ship
MSB	Most Significant Bit
NF	Number of Forwards
NORM	NACK Oriented Reliable Multicast
NTP	Network Time Protocol
OBU	On Board Unit
OS	Operating System
OSI	Open Systems Interconnection
PCF	Point Coordination Function
PDU	Protocol Data Unit
PHY	Physical
PMR	Professional Mobile Radio
PRoPHET	Probabilistic Routing Protocol using History of Encounters and Transitivity
PTPd	Precision Time Protocol daemon
Q-PRoPHET	Quality-PRoPHET
QCR	Quality Connection Reader
QoS	Quality of Service
RAM	Random Access Memory
RAPID	Resource Allocation Protocol for Intentional DTN
RF	Radio Frequency
RFC	Request for Comments
RSU	Road Side Unit

RTT	Round-Trip Time
Rx	Receiver
SN	Sequence Number
SSH	Secure Shell
SSI	Signal Strength Intensity
SSID	Service Set Identifier
SSP	Scheme-Specific Part
STA	Station
TCA	Tetherless Communication
TCP	Transmission Control Protocol
TCPCL	TCP Convergence Layer
TCS	Tetherless Computing Architecture
TETRA	Terrestrial Trunked Radio
UDP	User Datagram Protocol
UDPCL	UDP Convergence Layer
URI	Uniform Resource Identifier
USA	United States of America
UTC	Universal Time Coordinated
SBC	Single Board Computer
SDNV	Self-Delimiting Numeric Value
SDRAM	Synchronous Dynamic Random Access Memory
SV	Summary Vector
VANET	Vehicular Ad-Hoc Network
VDTN	Vehicular Delay Tolerant Network
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network

WSN	Wireless Sensor Network
XML	eXtensible Markup Language

Chapter 1

Introduction

1.1 Motivation

The navy operations include several elements (boats, helicopters, troops, etc.) that work between them with common objectives, and they need a communication system to synchronize and receive instructions during the operations. This communication system needs to guarantee reliable communications whenever possible.

Recently, the portuguese navy showed interest in investigating new communication technologies to trade their proprietary solutions by off-the-shelf devices with the use of ad-hoc network capabilities.

Figure 1.1 represents an aggregation of some navy scenarios where these communication networks will operate. In the left, there is a boat (with access to the Internet) responsible to coordinate the operation in the sea and shore, where there are several boats, and a helicopter acting as communication relay to the teams at the shore. The teams can also use the Internet access points to communicate with the coordinator boat.

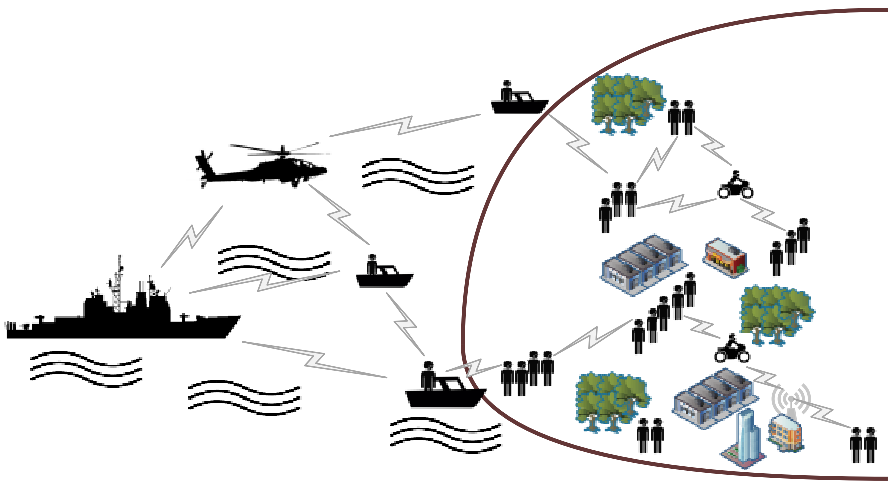


Figure 1.1: Navy Scenario

Usually the operations at the shore are ground recognition or population support. Independently of the operation, they are characterized by some issues such as: high mobility of the nodes, obstacles, reflections and interferences, which may create intermittent connections or even network fragmentation, which affect the navy communications. These constrains lead to low-quality communications that may be degraded due to bad routing decisions (i.e. the bundle is not sent to the best forwarder); this shall not happen because the communication is an essential factor that affects directly the success of the operations.

In Figure 1.1 the majority of the teams at shore have usually more than one path to communicate with the coordinator (left boat). Thus, an efficient routing algorithm should be capable to work with the presented constrains and select automatically the best quality communication path, whenever possible. The navy is also interested in using the Internet infrastructured accesses to communicate when they are present, be able to choose the best communication paths, and keep and store the information when no path is available.

Thus, the motivation of this work is to improve the communication in navy scenarios by offering more value with lower costs by exploring the possibility to use communication solutions that can be applied into off-the-shelf devices to this type of scenarios. These communications will use Delay Tolerant Networks (DTNs) as the communication base, because they are adequate to work with network fragmentation, due to their delay-tolerant mechanisms (store, carry and forward). To deal with the intermittent connections, it is proposed a quality-based routing algorithm to select the best quality paths to forward the information.

Although there are several DTNs solutions in the literature, a quality-based approach is not available, and this will be the aim of this Dissertation.

1.2 Objectives

The objective of this dissertation is to develop a routing protocol to DTNs based on the links quality with the capability to work in navy scenarios. The main objectives are as follows:

- Implement a modular method to gather physical information related with the links quality in off-the-shelf devices and extract this information to the routing and application layers directly from Physical (PHY) and Media Access Control (MAC) layers, in order to use them to other tasks that need them, e.g. a routing protocol that needs information on the links quality.

- Propose a routing protocol to DTNs based on the quality of the links, previous gathered, to deal with the navy constrains, such as intermittent connections.

- Show that DTNs can be applied to navy networks, using IBR-DTN with Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET). Show that DTN performance can be increased using a routing algorithm based on the quality of the wireless links, using the proposed routing algorithm in IBR-DTN.

- Assess the proposed routing protocol to perform routing in military scenarios using the quality links approach.

1.3 Contributions

The work developed in this dissertation has the following contributions:

- Description of methods to gather information about the PHY and MAC layers, related to the physical conditions of the links.
- Approach to define the quality of the links and a proposal of a modular Application Programming Interface (API) (Quality Connection Reader (QCR) module) to measure the quality of the relevant neighboring links and to easily integrate with DTN platforms.
- A proposal of a routing protocol to DTNs based on the quality of the links inspired on PRoPHET protocol, the Q-PRoPHET. At the best of my knowledge this is the first pure links quality-based routing protocol that exists for DTNs.
- Integration of Quality-PRoPHET (Q-PRoPHET) in an available DTN platform, the IBR-DTN.
- Writing of two papers about the work developed in this dissertation. The first one deals with the proposal of the quality-based measurement module and a first approach to routing in DTN, and it was published in *Conftele 2015*. The second one deals with the overall routing and evaluation, and it was submitted to the *IEEE International Conference on Communications 2016* (Mobile and Wireless Networks).

1.4 Document Organization

This document is organized as follows:

- **Chapter 1** contains the Introduction of the work.
- **Chapter 2** presents the state of the art about DTNs, Routing and the Standard IEEE 802.11, including also a brief overview about the Linux Wireless Subsystem and a description of methods to gather information directly related with the PHY layer.
- **Chapter 3** presents the navy scenarios for communication, the definition of quality of the links, and the proposed solution to perform routing in these scenarios using the links quality.
- **Chapter 4** presents the architecture and the implementation of the proposed solution.

- **Chapter 5** presents the integration and the evaluation of the implemented solution.
- **Chapter 6** presents the conclusions and the future work.

Chapter 2

State of the art

2.1 Introduction

This chapter is focused on providing the reader an overview of fundamental concepts needed to understand the work presented in this dissertation and also to present related work on the main topics.

Section 2.2 presents the concept of DTN. A brief overview and a definition of DTN is presented. Therefore, some applications of DTNs are presented as well as the DTN architecture together with the bundle protocol. Finally, some implementations of DTNs are presented.

Section 2.3 presents concepts about routing in DTNs. Finally, it presents work and studies related to routing in military networks.

Section 2.4 focuses on a brief description of IEEE 802.11 Wireless Local Area Network (WLAN) standard, the general architecture, the MAC layer and how it is implemented in the Linux Wireless Subsystem. Finally, it is presented the techniques to gather information, located in MAC and PHY layers, from the higher layers, due to their relevance regarding the work developed in this dissertation.

Section 2.5 presents the chapter summary and considerations.

2.2 Delay Tolerant Networks

2.2.1 Overview

The Transmission Control Protocol (TCP)/Internet Protocol (IP) model is very successfully nowadays in the actual Internet due to its efficiency, flexibility, robustness and the fact that it provides interoperable communications between all types of hardware and Operating Systems (OSs) [1]. The TCP/IP model allows the connectivity between several applications around the world serving many users. However, this model does not work well in certain circumstances, mainly in Mobile Ad-Hoc Networks (MANETs) where the end-to-end (e2e) connectivity can be non-existing or the connection can be very intermittent.

In these circumstances the TCP breaks [2] because it establishes a logical connection in an e2e path that it is not always available.

The main alternative to TCP/IP is the User Datagram Protocol (UDP)/IP model where the essential difference (not only one) is the absence of the logical connection. This way the source node can send immediately the data, but due to the fact that the e2e path may not be available, the data cannot be delivered to the destination. This way there is the need to create/use networks capable to operate under these circumstances. This type of networks are Delay Tolerant Networks (DTNs).

In reality the birth of DTNs (named by Kevin Fall of Intel Research Labs, Berkeley) was in August 2002 when Interplanetary Internet Research Group (IPNRG) published an updated version of the draft *Delay-Tolerant Network Architecture: The Evolving Interplanetary Internet* [3]. This draft describes a generalization of an architecture for Interplanetary Internets (IPNs), a communication system envisioned to provide Internet services across interplanetary distances in support of deep space exploration [4] . A more detailed description about the DTNs story and its relation with IPNs may be found in [5].

Throughout this section the DTN specifications are presented in more detail together with some implementations.

2.2.2 Definition

DTNs are networks that allow communications in challenging environments which are characterized by connectivity issues such as long and variable delays, sparse and intermittent connectivity, asymmetry data rates, packet losses, high errors rates and even nonexistence of e2e connectivity [5] [6] [7].

In DTNs the idea is that an e2e connection path may never be present. This way, to make a connection possible, the intermediate nodes should accept the data being transfered and store it. These intermediate nodes should forward the data when the opportunity arises, i.e. when another node appears or the destination is reached.

A real example to understand the DTN definition is DakNet [8], a project implemented in some villages in India and northern Cambodia. The concept of DakNet is that physical transports, e.g. buses, carry a mobile Access Point (AP) between villages' kiosks (without direct Internet access) and the city that has a Hub to the Internet, as shown in Figure 2.1. Data is automatically uploaded or downloaded when the bus is in the range of a kiosk or the Hub.

2.2.3 Applications

As previously stated, DTNs were initially proposed to IPNs to compensate the disconnections over interplanetary distances. However, DTNs can be applied in many challenging scenarios, which may oppose to one or more assumptions assumed by the e2e TCP/IP model. According to [9] this type of scenarios can be classified as:

- **Terrestrial mobile networks** may be partitioned due to nodes mobility and/or

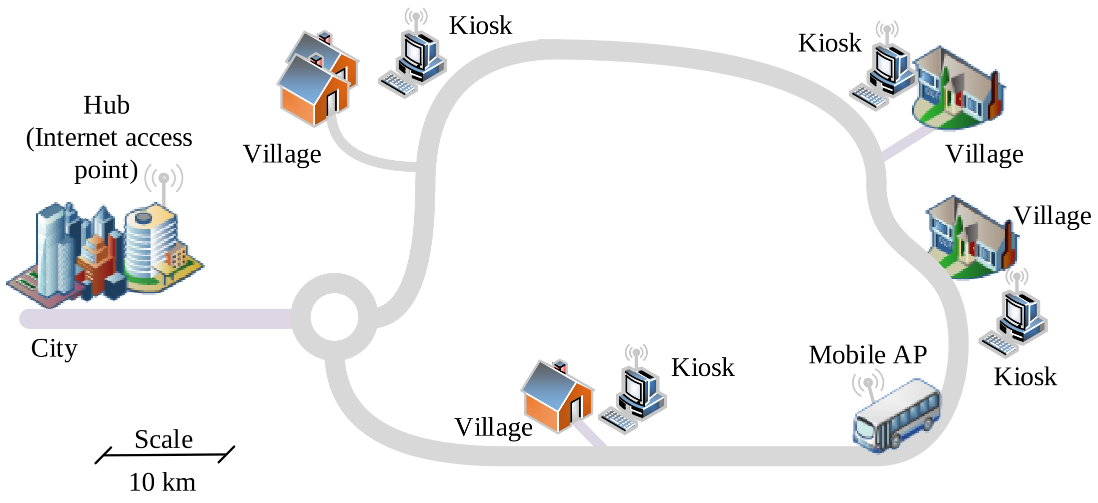
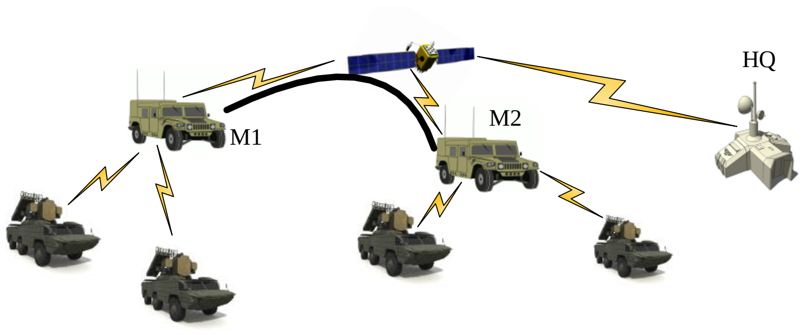


Figure 2.1: DakNet Concept - based on [8]

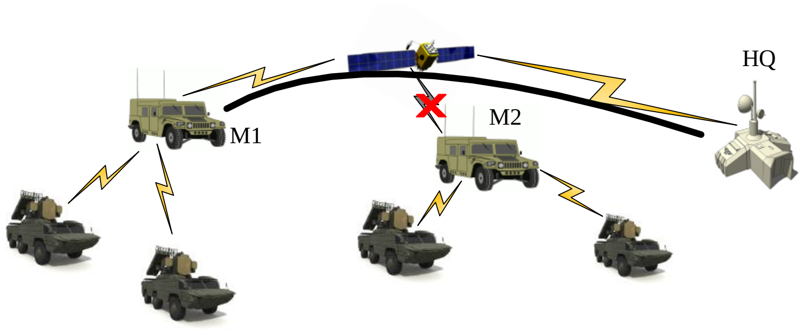
changes in Radio Frequency (RF) signal strength. This way the nodes need to store data until they reach other node that are suitable for delivering it to the destination. One good example of terrestrial mobile networks are Vehicular Delay Tolerant Networks (VDTNs) which extend Vehicular Ad-Hoc Networks (VANETs) with DTN capabilities to support long disruptions in network connectivity [10].

- **Exotic media networks** include near-Earth satellite communications, long distance links (either radio or optical), acoustic links in air or water and certain communications in free-space. As example, one satellite orbiting the earth may provide a predictability-available store and forward network service that is occasionally available when the satellite passes periodically by DTN nodes, possibly static in earth surface [9].

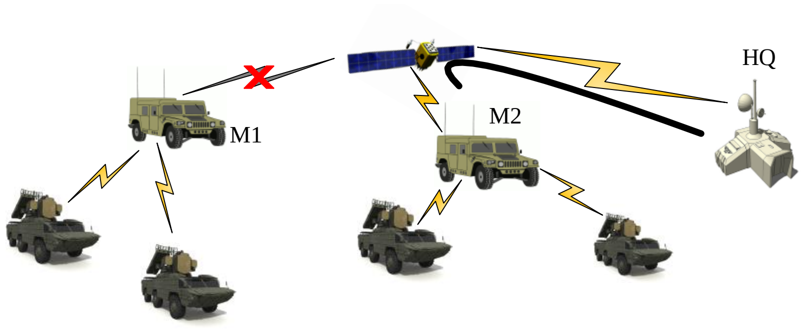
- **Military Ad-Hoc networks** should operate in hostile environments which consider mobility, environmental factors or intentional jamming that cause disconnections in the network. This way, the capability to store, carry and forward is an useful characteristic to serve the challenges of Military Ad-Hoc networks in hostile environments. Figure 2.2 shows an example of DTNs operation in a military scenario [11]. Mobile nodes M1, M2, and Headquarters (HQ) communicate with each other via satellite communication system. The connection between the satellite and HQ is often reliable. Firstly, in (1) M1 communicates directly with M2. However, due to various reasons, the link between M2 and satellite is down (2), and M1 needs to communicate with M2. DTN technology can be used to achieve the objective. M1 will route its data to HQ. Meanwhile the connection between M1 and satellite is down and the connection between M2 and satellite is restored. HQ realizes that M2 is available and transfers the stored data to M2 (3).



(1) M1 communicates directly with M2 via satellite



(2) the link between M2 with satellite is interfered, HQ store the data temporarily



(3) the link between M2 and satellite is restored, HQ forward data to M2

Figure 2.2: DTN Communication in military scenario - based on [11]

- **Sensor/Actuator networks** are a type of networks where the nodes are usually characterized by extremely limited end-node power, memory and Central Processing Unit (CPU) capabilities. In addition, they are supposed to exist in a large scale with thousands or millions of nodes per network. For this reason the communication may be scheduled in order to conserve power, or techniques may be adopted to decrease the number of active nodes, i.e. the nodes with communication capabilities activated. For example, the *Span algorithm* [12] is a power saving method applicable in wireless networks that reduces the energy consumption due to the fact that a dense network (many nodes per area unit) does not need to have all nodes with wireless capabilities activated, since only some nodes are enough to guarantee multi-hop packet routing. This algorithm selects coordinators that stay awake to perform routing, and these coordinators vary along the time to distribute the energy consumption.

2.2.4 Architecture

The DTN architecture [13] is based on the insertion of a new layer above the transport layer (or others), the bundle layer. The comparison between Open Systems Interconnection (OSI) layers, TCP/IP layers and DTN layers is presented in Figure 2.3. The bundle layer can handles with delays and disruptions at each DTN node in a path between the sender and the destination. Nodes on the path can provide network capabilities to store, carry and forward data (usually called bundle(s) in DTNs) using the network layers specific to each DTN region, often called convergence layers.

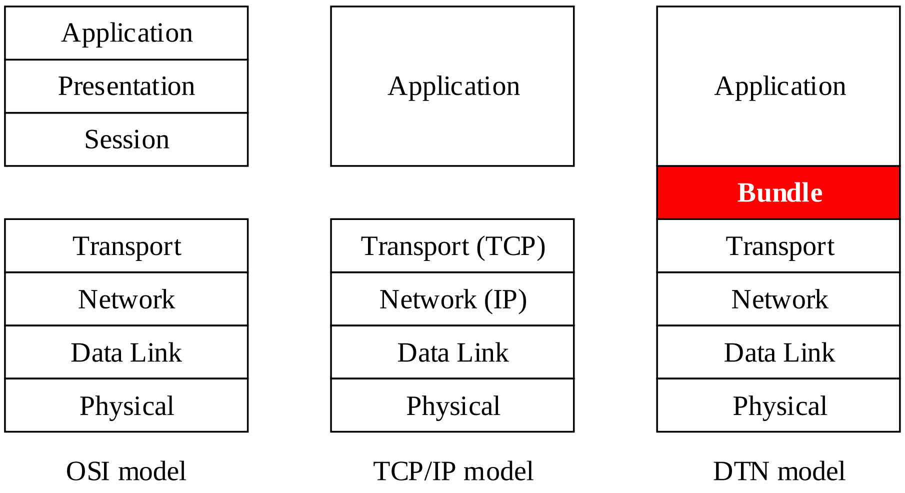


Figure 2.3: Bundle layer

A high-level conceptual implementation of DTN architecture at bundle layer is presented in Figure 2.4. This architecture comprises a central forwarder that is responsible for moving bundles between applications, Convergence Layer Adapters (CLAs) (to communicate with underlying networking layers) and persistent storage, according to routing decisions [6]. The arrows indicate interfaces which can transport either bundles or directives (routing decisions, management, applications).

The following subsections explore in more detail the DTN architecture [13] [14].

2.2.4.1 Virtual Message Switching Using Store-and-Forward

A local DTN-enabled application can send and/or receive messages of arbitrary length, usually called Application Data Units (ADUs) to/from bundle layer. The ADUs are transformed at bundle layer in one or more Protocol Data Unit (PDU), usually called Bundles.

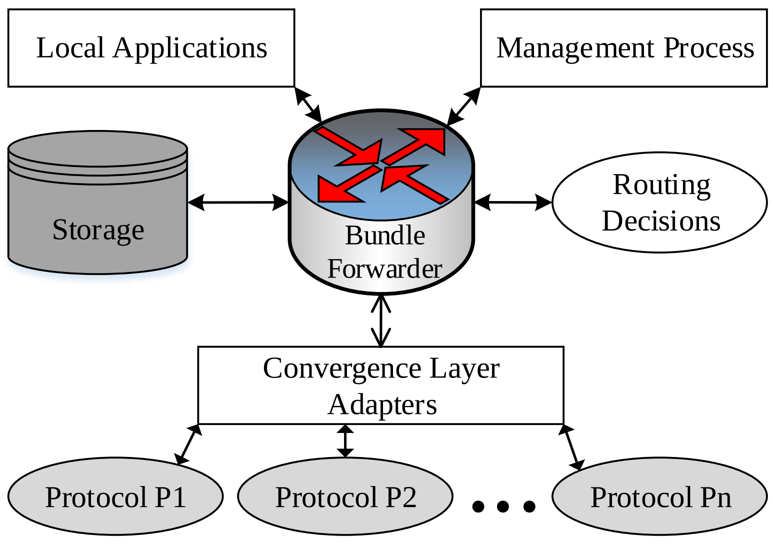


Figure 2.4: DTN Conceptual Architecture - based on [6]

The bundles have a specific format containing two or more blocks of application data or control information to forward and deliver the bundle to its destination.

Unlike the IP networks based on the assumption that ”storing” will not persist for more than a small amount of time, the DTN does not assume the same because it does not expect that the connections to other nodes are always available or reliable. Therefore, the most important assumption of DTN forwarding is that bundles should have a place to wait at nodes until a communication opportunity is available. This assumption assumes that storage is always available and well-distributed over the network, as well as sufficiently robust to store as many bundles as necessary. Due to the fact that the previous assumption is not always true the research in DTNs revolves around exploring this issue in many ways, but essentially in improving and creating new routing algorithms capable of forwarding the bundles in a fast and efficient manner.

2.2.4.2 Nodes and Endpoints

A DTN node is an engine for sending, forwarding and receiving bundles, i.e. nodes with a bundle layer attached. Applications can use these nodes to send and receive ADUs.

DTN nodes may belong to one or more groups called DTN endpoints which is a set of DTN nodes. A specific bundle is considered successfully delivered to a DTN endpoint when a minimum group of nodes in the endpoint has received that bundle. This group is called Minimum Reception Group (MGR) and may be one node (unicast), a group of nodes (anycast) or all nodes (multicast and broadcast).

2.2.4.3 Endpoint Identifiers and Registrations

Endpoint Identifiers (EIDs) are names, with some rules, that identify DTN endpoints, and usually they are expressed using Uniform Resource Identifier (URI) syntax [15] [16]. An URI is a string that is comprised of a Scheme Part and a Scheme-Specific Part (SSP): <scheme>:<SSP>. Using an EID it is possible to determine the MGR of the DTN group named by that EID. However, each node needs to have a unique EID that identifies it in the network.

When an application aims to send an ADU to another node, it needs to know the EID from that node to be possible for DTN nodes in the network to operate and deliver the bundle to the destination.

When a local application aims to receive bundles from other nodes, it needs to demonstrate that intention. This intention is shown by performing a registration in the DTN node. This registration may or may not be persistent, depending generally if the application wishes to continue registered or not.

2.2.4.4 Routing and Forwarding

The DTN architecture (Figure 2.4) offers a framework for routing decisions and forwarding at the bundle layer. The DTN architecture [13] defines different categories of contacts in DTNs as follows:

- **Persistent Contacts** are always available, i.e. *always-on* like a good Internet connection. These contacts are the best case for DTNs.

- **On-Demand Contacts** need a trigger action to instantiate, but then they are like a persistent contact. A good example of an on-demand contact is the dial-up connection.

- **Intermittent - Scheduled Contacts** are contacts with an agreement between two or more nodes to establish a contact at a particular time with a certain duration. An example of a scheduled contact is present in Figure 2.1, if the bus has a precise fixed schedule.

- **Intermittent - Predicted Contacts** are, as the name says, expected contacts but they are not scheduled. An example of a predicted contact is presented in Figure 2.1 if the bus has a random schedule based on the driver's desire.

- **Intermittent - Opportunistic Contacts** happen by chance, they are not scheduled or predicted, but due to the fact that two or more nodes encounter themselves the contact is made. An example of an opportunistic contact is a smart-phone with Wi-Fi when it encounters an AP. It can synchronize the e-mail because it encounters an Internet access by chance.

There are many routing algorithms for DTNs. Some of them are presented in section 2.3, but each of them has assumptions that make it work well for a specific type of contacts only.

2.2.4.5 Bundle Fragmentation and Reassembly

The DTN architecture contains also a fragmentation module that has the capability to fragment and reassemble bundle fragments [17]. This module is designed to improve the efficiency of bundle transfers by guaranteeing that contact volumes are completely utilized, and by avoiding retransmission of partially forwarded bundles. There are two forms of DTN fragmentation and reassembly:

- **Proactive Fragmentation** occurs when an ADU is divided into multiple smaller blocks and each block is transmitted as an independent bundle by the DTN node. This way the destination is responsible for reassembling the smaller blocks into the original bundle and obtaining the original ADU. The name of this approach is related with the fact that fragmentation is made before sending the bundle(s). As example, the proactive fragmentation may occur when DTN nodes can predict the contact volumes in advance.

- **Reactive Fragmentation** occurs when an active bundle transfer is interrupted and only a fragment of the bundle was transfered. In this situation the receiver bundle layer changes the received fragment to indicate that it is a fragment and forwards it normally. The sender node may notice that only a fragment was transfered to the next hop, and sends the remaining bundle when the contact becomes available again. The name of this approach is related with the fact that fragmentation occurs after an attempted transmission was interrupted.

The reactive fragmentation capability is not a requirement that needs to be available in all DTN implementations, since not all underlying protocols support this and because it also presents significant challenges related to security.

Even with fragmentation disabled in a DTN node, the capability of reassembly should be active because the node may receive fragments from other nodes, and it needs to reassemble them in order to obtain the complete bundle.

2.2.4.6 Reliability, Custody Transfer and Security

The basic service offered by the bundle layer is unacknowledged, i.e. the nodes prioritize unicast message delivery, but they do not guarantee that messages will be delivered. However, the bundle layer also offers the custody transfer mechanism which is a service that may be provided to a specific bundle while it is crossing the network. The objective of custody transfer is to keep track of a current custodian for each bundle. The custodian is required and has the responsibility to keep the bundle secure and safe in the persistent memory until it delivers the bundle successfully to another custodian [6]. This way the responsibility to deliver the bundle passes through the DTN nodes with custody capabilities instead of the original sender.

Not all DTN nodes need to have custody support. A node can receive a bundle without accepting the responsibility to deliver it due to memory limitations or other reasons. Custody transfer mechanism may not be a true hop-by-hop mechanism because a bundle transfer between two custodians can occur through many DTN nodes without custody support.

However, to guarantee reliability and custody transfer between DTN nodes, security may be considered. Security is an important topic in networks nowadays and DTNs are not an exception. To define security in DTNs there is the *Bundle Security Protocol* [18] that describes the bundle security protocol and a set of mandatory *ciphersuites*[1]. The bundle security protocol defines rules to authenticate nodes, guarantee bundles integrity and confidentiality, transport security keys and also to deny access to unauthorized applications and prevent authorized applications from abusing the DTN to their advantage.

2.2.4.7 Timestamps and Time Synchronization

Bundles have associated to them a timestamp that indicates the creation time and that is used for identification purposes. Besides that, bundles' timestamps are also usefull for routing strategies with scheduled or predicted contacts, bundle expiration time computations and application registration expiration.

In terms of identification, the EID source, timestamp and data offset/length allow DTN nodes to identify uniquely each bundle, even if they are fragments.

In terms of routing, if the bundles contain the timestamp, some routing strategies may use this information to schedule sending the bundles to respective destinations before the bundles expiration time, that is usually calculated by adding the timestamp and the lifetime present in the bundle. However, if the lifetime of a specific bundle expired, the nodes may delete this bundle from storage because it is no longer useful and the node can free some persistent memory. When an application registers at the bundle layer, it can also express its desire to receive ADUs until a finite amount of time.

The last utilities presented are only viable if the DTN nodes are synchronized between them. If they are desynchronized, unwanted behaviors could occur. As an example, if a DTN node has its internal clock set in advance, when it receives a bundle, it will probably consider that the bundle is expired due to the fact that its clock has a significant offset. Thus, DTN nodes should have mechanisms to guarantee that nodes are synchronized between them. However, this feature does not require an exactly accurate synchronized time, but deviations of more than a few seconds could be problematic depending on the application.

There are many applications developed for offering time synchronization between nodes. Network Time Protocol (NTP) [19] has provided accurate synchronization within Internet from years. Precision Time Protocol daemon (PTPd) [20] offers a good synchronization in Ad-Hoc networks. Most existing networks for extreme environments already provide some (often out-of-band) means for obtaining accurate time [21].

[1] *"A ciphersuite is a specific collection of various cryptographic algorithms and implementation rules that are used together to provide certain security services."* [18]

2.2.5 Bundle Protocol

The bundle protocol [16] defines the bundle layer (Figure 2.4) that offers an e2e architecture supporting communications through highly challenging environments that include intermittent connectivity, large or variable delays and high bit error rates. To offer reliability in this type of networks, the bundle protocol is placed above the transport layer and under the application layer. According to [16], key capabilities of the bundle protocol are:

- Custody-based retransmission;
- Ability to lead with intermittent connectivity;
- Ability to take advantage of scheduled, predicted, opportunistic and continuous connectivity (See Routing and Forwarding in section 2.2.4.4 for more details);
- Late binding of overlay network EIDs to constituent Internet addresses.

The bundle protocol also defines the bundle's format. A bundle is formed by a series of contiguous data blocks where each bundle is formed by two or more blocks of protocol data, which serve various purposes. The bundles pass through the network in a store, carry and forward method between the DTN nodes over different network transport layers including TCP/IP and non TCP/IP transport/network layers [14].

The following subsections explore in a deeper way the bundle protocol.

2.2.5.1 Service Description

The bundle protocol at each node is expected to offer the following services to the node's applications that want to use DTN services:

- Initiate a registration (registering a node in an EID);
- Finishing a registration;
- Switching a registration between Active and Passive States;
- Transmitting a bundle to an EID;
- Canceling/Aborting a transmission;
- Verify a registration that is in a passive state;
- Deliver a received bundle.

2.2.5.2 Bundle Format

Each bundle is formed, at least, by two blocks that are concatenated. The first block should be the only Primary Bundle Block (Figure 2.5) in the bundle. Additionally different blocks may follow the Primary Bundle Block to support specific extensions, such as the Bundle Security Protocol [18]. Furthermore, at least one block should be a Payload Block (Figure 2.5) and the last bundle's block should have the *last block* flag (in processing control flags) set to '1'. All other non Primary Bundle Blocks should have this flag set to '0'.

Primary Bundle Block:

Version (1 byte)	Processing Control Flags (SDNV)
Block length (SDNV)	
Destination scheme offset (SDNV)	Destination SSP offset (SDNV)
Source scheme offset (SDNV)	Source SSP offset (SDNV)
Report-to scheme offset (SDNV)	Report-to SSP offset (SDNV)
Custodian scheme offset (SDNV)	Custodian SSP offset (SDNV)
Creation Timestamp time (SDNV)	
Creation Timestamp Sequence Number (SDNV)	
Lifetime (SDNV)	
Dictionary length (SDNV)	
Dictionary byte array (variable)	
Fragment offset (SDNV, optional)	
Total application data unit length (SDNV, optional)	

Bundle Payload Block:

Block type (1 byte)	Processing Control Flags (SDNV)	Block length (SDNV)
Bundle Payload (variable)		

SDNV = Self-Delimiting Numeric Value
SSP = Scheme-Specific Part

Figure 2.5: Bundle Block Formats - based on [16]

The bundle protocol was designed to minimize the consumption of transmission bandwidth and to allow extensibility of address requirements that are not yet identified, as well as scalability through a wide range of network scales and payload sizes. For this purpose the Self-Delimiting Numeric Values (SDNVs) was proposed to encode bundle block fields with variable length. SDNV is an encoding scheme adapted from [22] [23] and it is a numeric value encoded in N octets, where all octets have their Most Significant Bit (MSB) set to '1', except the last octet that has its MSB set to '0'. The values encoded in SDNVs are obtained by extracting the least 7 bits from each SDNV and concatenating them to form a single bit string [16]. Figure 2.6 presents an example of an encoding process to a hexadecimal value.

0xABD	:	1010 1011 1101
		is encoded as
		[1 00 10101] [0 0111101]
		10010101 00111101

Figure 2.6: SDNV encoding example

The fields of the Primary Bundle block, presented in Figure 2.5 are:

- **Version** indicates the version of the bundle protocol that built this block;
- **Processing Control Flags** contains flags that are used for status reports, class of service and general information relative to the bundle;
- **Block length** contains the length of all remaining fields of the block;
- **Destination scheme offset** contains the dictionary offset referencing the byte array of the scheme name of the EID of the bundle's destination;
- **Destination SSP offset** contains the dictionary offset referencing the byte array of the SSP of the EID of the bundle's destination;
- **Source scheme offset** contains the dictionary offset referencing the byte array of the scheme name of the EID of the bundle's nominal source, i.e., the initial source EID that it initially transmitted;
- **Source SSP offset** contains the dictionary offset referencing the byte array of the SSP of the EID of the bundle's nominal source;
- **Report-to scheme offset** contains the dictionary offset referencing the byte array of the scheme name of the Identification (ID) of the endpoint to which the status report should be transmitted;
- **Report-to SSP offset** contains the dictionary offset referencing the byte array of the SSP of the ID of the endpoint to which the status report should be transmitted;
- **Custodian scheme offset** contains the dictionary offset referencing the byte array of the scheme name of the EID of the current bundle's custodian;
- **Custodian SSP offset** contains the dictionary offset referencing the byte array of the SSP of the EID of the current bundle's custodian;
- **Creation Timestamp** is used for bundle identification together with the source EID and (if the bundle is a fragment) the fragment number offset and payload length. The creation timestamp contains the bundle's creation time (in seconds since year 2000

on Universal Time Coordinated (UTC) scale [24]) and the sequence number, which is a positive integer number that is increased for each bundle creation and is reseted whenever the current time advances by one second in order to distinguish different bundles created at the same node at the same second.

- **Lifetime** indicates the bundle's useful lifetime, i.e., the time in which the bundle's payload is no longer necessary expressed in seconds after the creation of the timestamp time. This field is useful for nodes to know when they do not need to store or forward the bundles in order to delete them from the network.
- **Dictionary length** contains the length of the dictionary byte array field;
- **Dictionary byte array** is a byte array formed by all schemes and SSPs pointed by previous dictionary offsets;
- **Fragment offset** indicates the fragment number if the bundle is a fragment. If the bundle is not a fragment this field does not appear in the block.
- **Total application data unit length:** If the bundle is a fragment this field indicates the total length of the original ADU of which this bundle is a fragment. If the bundle is not a fragment this field does not appear in the block.

The fields of Bundle Payload Block, presented in Figure 2.5 are:

- **Block type** indicates the block type. For a payload block, this field has the value '1'.
- **Processing Control Flags** contains seven flags (bits) to indicate some information relative to the block. For example, bit 3 indicates that the block is the last block from the bundle.
- **Block length** contains the total length of remaining field of the block, i.e., payload length;
- **Bundle Payload** contains the ADU.

2.2.6 Implementations

2.2.6.1 DTN2

DTN2 [25] is the reference implementation of the DTN Bundle Protocol [16] and its main goal is to provide a flexible and robust software framework for experimentation, extension and real-world deployment. It is written in C++ and uses a framework called Oasys that was designed to provide an uniform interface to DTN2, hiding the OS and other support package details.

The architecture of DTN2 daemon is presented in Figure 2.7 and is composed by the following blocks:

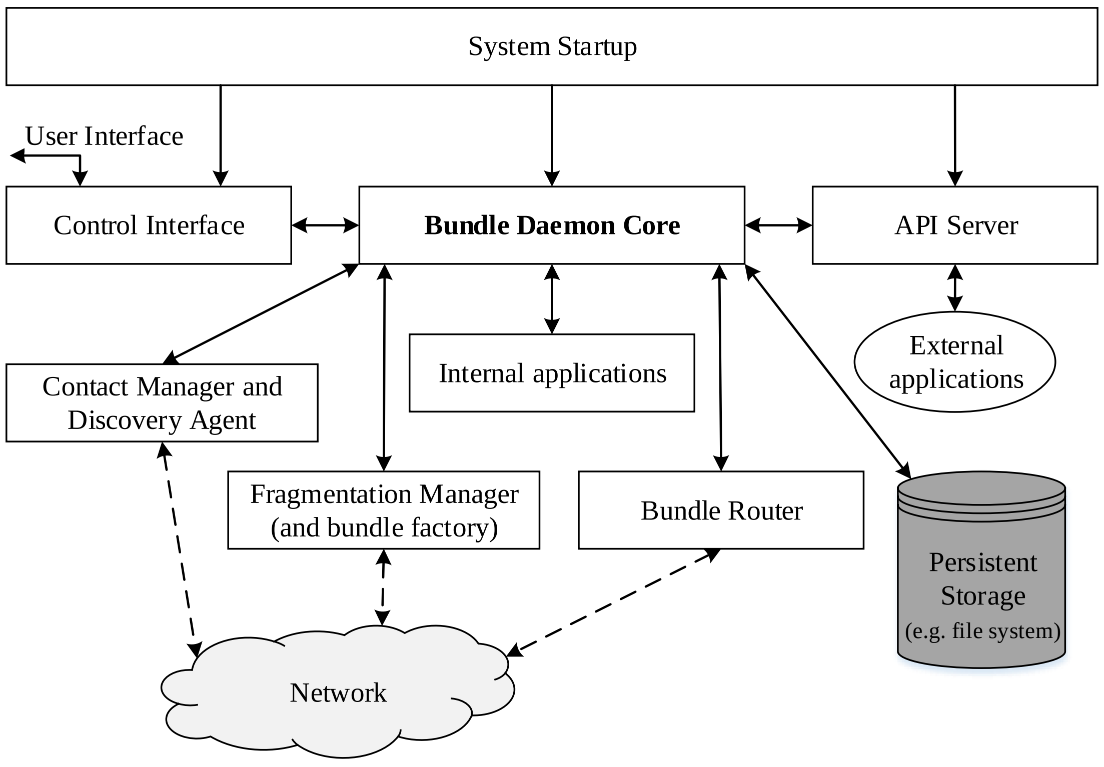


Figure 2.7: DTN2 architecture - based on [26]

- **System startup** that manages the begin of all components and the creation off all relevant threads (Bundle Daemon Core, API Server and User Control Interface).
- **API server** is the interface between daemon and applications. It deals with application registrations and has the capability to receive/deliver bundles from/to applications. It also supports a publish and subscriber mechanism that is not fully documented.
- **Control Interface** provides a way to configure and monitoring Daemon operations. It is an interface that can receive commands and may present data to the user.
- **Bundle Daemon Core** is the central component of the daemon architecture. It acts like an event dispatcher for the rest of the system and it is not a state machine, because all events can be handled at any time. According to [26] Bundle Daemon Core manages:
  - the network interfaces in local node;
  - sending and receiving node advertisements on interfaces relative to the DTN node discovery;
  - opening and closing links to other nodes;

  - the bundle router that selects and schedules bundles for transmission on network interfaces;
  - storing and retrieving bundles to/from local node's persistent storage;
  - accepting bundles or delivering them from/to local applications according to the application registrations;
  - the timers used for various purposes, e.g. bundle expiration.

- **Fragmentation Manager** (and bundle factory) deals with operation of split bundles (either reactively or pro-actively) or organizing sets of fragments of a bundle, and determining when the available fragments are enough to form an entire bundle so that it can be reassembled and transmitted.

- **Persistent Storage** is a file system or database based storage where the bundles can be preserved across daemon or node restarts.

- **Bundle Router** is responsible to make all decisions relative to routing and forwarding bundles. It receives events from the Bundle Daemon Core and, based on them, it decides when it should forward bundles that are stored locally. DTN2 includes several routing modules, which are:

  - **Static routing** where all routes and links are configured statically, i.e., the *next-hop* routes are created and removed by configuration commands;
  - **Epidemic routing** is a variant of flooding routing. A more complete description is presented in section 2.3.2.1.
  - **PRoPHET** is an implementation of Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET). A more complete description is presented in section 2.3.2.2.
  - **Delay Tolerant Link State Routing (DTLSR)** is a link state based routing protocol for DTNs. DTLSR [27] realizes network state changes due to the flooded link state announcements throughout the network. All nodes maintain graphs (one graph for each node) representing their current view of the network and they use shortest path computation (e.g. *Dijkstra*) to perform routing decisions often based on an estimation of the delay that it would take to send a message using the given link. However, each node maintains graphs only to its administrative area which helps to contain the size of the network graph and limit the scope of the routing messages. Neighbors in other administrative areas are reachable via a set of EIDs that announce themselves as gateways for other neighbors.
  - **Tetherless Computing Architecture (TCS)** routing [28] is an opportunistic-based protocol used mainly to allow communications between different DTN regions. It uses a centralized database (usually in the Internet) that knows the region of each DTN node that is reachable via the region's gateways (DTN

routers that connect more than one region with connection with the centralized database). When a certain node needs to communicate with another node in another region, the bundle passes though the source's region until the gateway, and then it is forwarded to the destination's region in order to reach the destination according to the informations present in the centralized database.

- **Contact Manager and Discovery Agent** has the responsibility to manage communication links to other nodes. These links can be used to send and receive bundles to/from other reachable DTN nodes. Each link has a respective convergence layer which determines the lower level transport protocol which should be used to support the communication. The links can be created due to administrative management requested either from a configuration file or during operation; as a consequence of an advertisement received from a neighbor discovery mechanism or as a consequence of a connection request received in a associated convergence layer from another node.

- **Internal applications:** DTN2 provides several applications with its distribution. *Dtnping* is an application that sends bundles to an EID and waits for an answer. The response time is measured and is printed out. *Dtnsend* is used to create a bundle with data and send it through the DTN, and *dtnrecv* is used to retrieve the bundle out in the final destination. *Dtncp* is used to move a file from a source node to a destination node. However, in the receiving node a presence of *dtncpd* is necessary, a daemon registered with local bundle router that requests notifications when a bundle comes in that was sent by dtncp. If it receives a bundle, it will put the bundle into a specified directory.

2.2.6.2 IBR-DTN

IBR-DTN [29] was developed in Institut für Betriebssysteme und Rechnerverbund (IBR) at Germany. It is considered a lightweight, modular, efficient and highly portable implementation of Bundle Protocol [30] [31]. IBR-DTN was created specially for embeded systems, in C++, with limited constrains in terms of memory and processing. It is compatible with several convergence layers, such as TCP, UDP and others.

IBR-DTN daemon architecture is presented in Figure 2.8 and it is composed by the following blocks:

- **Event Switch** is the IBR-DTN core. This module dispatches a set of standard events to all relevant sub-modules. It acts as an event's forwarder, but events that trigger high processing can be queued directly in a private queue of the module's thread, which allows a high level of concurrency between modules. This way the existing modules, and the new ones, can receive and raise events to communicate with other daemon's modules.

  The standard events exist to deal with bundles routing, storage operation and nodes availability.

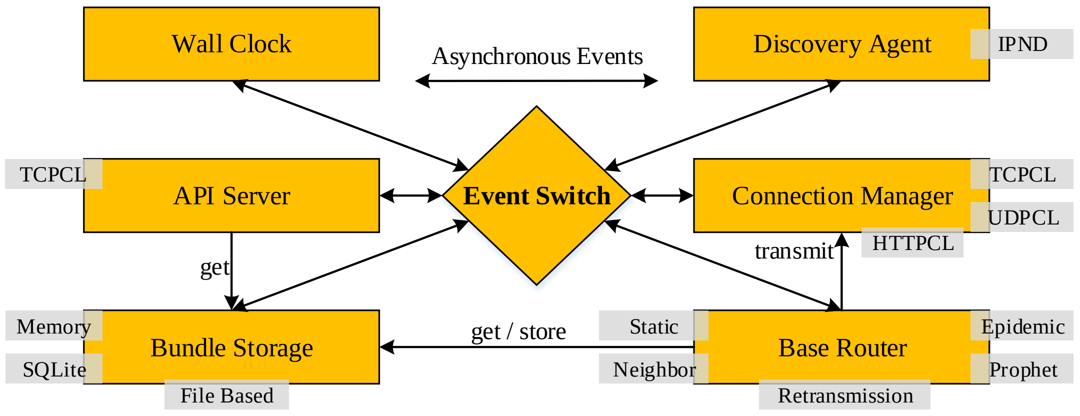


Figure 2.8: IBR-DTN architecture - based on [31]

- **Discovery Agent** has the function to discover other nodes through pluggable discovery modules. It uses DTN IP Neighbor Discovery (IPND) (v1 and v2) specified in [32] and it is compatible with DTN2 IP-Discovery frames. Discovery Agent creates events related to appearance or disappearance of the neighbors. Whenever a neighbor is detected, the routing modules will verify if there are new bundles that should be transmitted to the new neighbor.

- **Connection Manager** manages the instances of convergence layer modules. Each convergence layer offers an interface to transfer bundles to the neighboring nodes. Incoming bundles are also announced by a global event and stored in the bundles storage.

  IBR-DTN has four built-in convergence layers:

  - **TCP Convergence Layer (TCPCL)** is compatible with Delay-Tolerant Networking TCP Convergence-Layer Protocol [33] and uses a handshake mechanism between different DTN daemons, including also the ability to split bundles into segments that are acknowledged by receiving daemons.
  - **UDP Convergence Layer (UDPCL)** is compatible with Datagram Convergence Layers for the Delay- and Disruption-Tolerant Networking (DTN) Bundle Protocol and Licklider Transmission Protocol (LTP) [34]. It offers a very simple way for transferring bundles between daemons due to the fact that it uses UDP, a simple transport protocol. UDPCL requires that a bundle fits into a unique UDP datagram, which the maximum bundles size is limited.
  - **HTTP Convergence Layer (HTTPCL)** is a convergence layer based on *libcurl* and can use an Hypertext Transfer Protocol (HTTP) server to send or receive bundles.
  - **LowPAN Convergence Layer (LowPANCL)** supports 802.15.4 MAC protocol [35], commonly used in Wireless Sensor Networks (WSNs).

- **Bundle Storage** module is what allows the DTN nodes to store bundles for extended periods of time. Bundle Storage module offers an interface to store and retrieve bundles by different search criteria, e.g. by bundle ID. There are three types of bundles storage and one type can be selected in the configuration file. The different types are listed below:
  - **Memory** is a volatile storage wherein all bundles are kept in Random Access Memory (RAM) and it is the default type of storage in IBR-DTN. A maximum size limit can be defined.
  - **File Based Storage** is a persistent storage type based on simple files stored on the hard disk. In this type the bundles survive daemon restarts and power failures of nodes. Since the bundles are not in memory, the requirements in terms of memory size are reduced.
  - **SQLite** uses a SQLite database as backend. It can store higher amounts of meta information for bundles which is useful for complex routing algorithms.
- **Base Router** is responsible for managing different routing modules, which can work concurrently. It receives events relative to arriving or departing nodes from the Discovery Agent. They are also notified whenever new bundles arrive in storage. If a certain routing module considers itself responsible for a given bundle, it can notify the Connection Manager and uses the appropriate convergence layer to transfer the bundle to the next node. IBR-DTN includes several routing modules, which are:
  - **Static** where routes and links are configured statically, i.e., the convergence layer for a given EID is configured *a priori* in the configuration file. This assumes that static routes are always available.
  - **Neighbor** routing module just routes bundles to nodes discovered by the Discovery Agent, i.e., the direct neighbor nodes. When using IPND, the nodes present in the same subnet are reachable.
  - **Retransmission** routing module is used when a convergence layer signals an error while it was trying to transmit a bundle to the next hop. In this case the bundle is re-queued and continues at storage. When possible, the Retransmission module will try to repeat the transmission to the next hop.
  - **Epidemic** routing is implemented also in IBR-DTN. A more complete description is presented in section 2.3.2.1.
  - **PRoPHET** is an implementation of Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET) protocol whose description is presented in section 2.3.2.2.
- **Wall Clock** module determines the current global time in the DTN by evaluating the local host's clock. The timestamps in the Bundle Protocol count the time in seconds since January 1, 2000. In addition, this module is also used to initiate recurring tasks.

- **IBR-DTN API** is a socket based API interface: it can be a TCP socket which allows users to run daemon and API on different machines, or it can be a Unix Domain Socket if the daemon and API are running at the same machine.

  To avoid reimplementing the complex bundle streaming protocol in each DTN application, IBR-DTN offers a library that can be linked with user's applications simplifying the creation of bundles.

- **Tools:** IBR-DTN provides several tools with its distribution [29]. *dtnsend* and *dtnrecv* are two applications to send and receive files. *Dtnping* is an application that sends bundles to an EID and waits for a bundle with the same payload as reply. The response time is measured and it is printed out. *Dtntracepath* prints the path of a bundle through the network using "bundle forwarded" reports from all hops. *Dtntunnel* is an experimental tunnel that encapsulates IP traffic through a DTN connection. *Dtninbox* and *dtnoutbox* are two applications to receive/send files from/to two directories on different nodes. There is also *dtntrigger* which is a lightweight alternative to the API and can be used to receive bundles and execute programs.

2.2.6.3 Helix

Helix DTN was developed by researchers from Instituto de Telecomunicações of Aveiro and Veniam® [36], a technological company that develops wireless technologies to vehicular networks.

Helix is a DTN platform designed specially to wireless networks and it meets the requirements to work specially in vehicular networks. Helix distinguishes types of nodes according to their functionality in the network, e.g. the cars/trucks/bus are On Board Units (OBUs), static sensors, the fixed accesses to infrastructure are Road Side Units (RSUs) and these generally are connected via Internet to a Server. This way it is possible to perform routing according to the type of the nodes and their associated behavior.

Helix does not comprise all rules defined in the RFCs [13] and [16] because the objective is to have a simple DTN with low resource consumption and low overhead. For example, in Helix the nodes are identified with a number (integer) and not with a string that requires generally more bytes to store it, increasing the overhead in the network.

Helix was specially developed to collect data from sensors and logs from Veniam's vehicular boards, and also to perform content distribution, i.e. software updates for the boards, commercial advertisements or entertainment content. All applications that require a DTN in vehicular networks may use Helix.

The Helix architecture is presented in Figure 2.9.

Helix is composed by the following blocks:

- **Socket** module is an UDP socket and it works as an abstraction layer to send and to receive Helix packets to or from the neighbors, respectively.

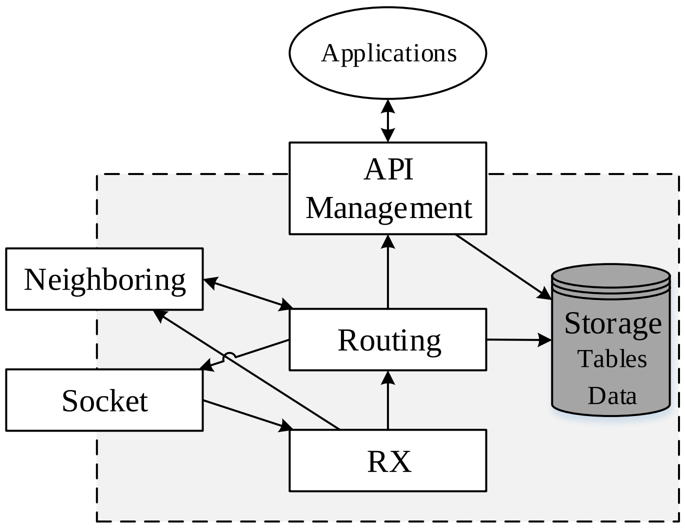


Figure 2.9: Helix architecture

- **Receiver (Rx)** is a module that queries the socket for new packets. If they exit, it receives the packets and analyzes and classifies them according to their headers, and it decides if it should forward or drop the packets.
- **Neighboring** module listens the media searching from other nodes running Helix, i.e. neighbors, and manages information relative to the interface where the neighbors are seen.
- **Storage** module stores information in persistent memory and in volatile memory. The bundles are entirely stored in persistent memory, but some information about the bundles, that needs to be accessed in a fast way, is maintained in RAM.
- **Routing** module is responsible to manage the bundles. It decides which bundles should be sent and in which order and to which neighbors, according to the routing strategy.
- **Applications** contain some applications that can use Helix DTN. The current available applications are `HelixPing` that sends a packet to other DTN node and waits for the reply and measures the Round-Trip Time (RTT). `HelixSendString` and `HelixRecvString` are two applications to send and to receive a string between two nodes, respectively; `HelixSendFile` and `HelixInbox` are two applications to send and to receive files between DTN nodes.

2.2.6.4 Comparison between DTN implementations

Table 2.1 presents a high level comparison between the DTN implementations presented in section 2.2.6. The comparison focuses in the main characteristics of each implementation.

Table 2.1: Comparison between DTN implementations

DTN Implementation	OS	Programming Language	Security Support	Applications	Routing	Observations
DTN2 [25] [26]	Debian/Ubuntu, MacOS X, Solaris, FreeBSD and Linux on ARM	C++	OpenSSl and partial support for BSP [18]	dtnping, dtnsend, dtnrecv, dtncp and dtncpd	Static, Epidemic [37], Flooding, PRoPHET [38] [39], DTLSR [27], TCA [28], external routing via XML	It is the general reference implementation for DTNs. It presents faster transmission than IBR-DTN [40] and it presents scalability issues [41].
IBR-DTN [29] [31] [30]	OpenWRT, Debian/Ubuntu, Debian ARM, MacOS X, Gentoo Linux, Windows and Android	C++	Combination of 4 levels (none, authenticated bundles, encrypted bundles, signed bundles) based on [18]	dtnsend, dtnrecv, dtntrigger, dtnping, dtntracepath, dtninbox, dtnoutbox and dtnstream	Static, PRoPHET [38] [39], Epidemic [37] and Flooding	It is a general implementation for DTNs. It is considered an efficient DTN implementation and consumes few memory resources when compared with DTN2 [30] [31] [40] [42].
Helix	OpenWRT, Debian/Ubuntu	C++		HelixPing, HelixSendString, HelixRecvString, HelixSendFile and HelixInbox	routing_v0 and routing_v1 (based on the type of the nodes)	It is developed specially for VDTN

By analyzing table 2.1, it is possible to verify that all implementations run in OpenWRT which is the OS that is used in the evaluation of the work (section 5.3); the embedded systems and implementations are written in C++. In terms of security there are differences, but the objective of the work is related with routing and not with security. In terms of native applications, IBR-DTN offers more applications than the others and with the same or more functionalities.

Relatively to performance metrics, according to *Doering et al.* [30], IBR-DTN is an efficient implementation for embedded systems and it consumes few memory resources. *Beuran et al.* [41] and *Georgescu et al.* [40] made some performance evaluations between IBR-DTN and DTN2 and in summary they conclude that DTN2 presents better transmission than IBR-DTN, i.e. in good conditions of communication DTN2 was capable to transfer more data than IBR-DTN. However, *Beuran et al.* and *Georgescu et al.* also verified that DTN2 has scalability issues and it has higher CPU utilization and higher memory consumption than IBR-DTN, which is a big disadvantage for DTN2 to run in embedded systems with low memory and CPU resources.

Due to the stated reasons and the fact that Helix is destined to VDTNs, the IBR-DTN was selected as the base implementation of DTNs to be used in this dissertation.

2.3 Routing

2.3.1 Overview about Routing

Routing is the process of selecting a path to send data to one destination (or more) in a network. Usually, the idea is to have a routing algorithm, running at the nodes, capable of selecting the best path to a destination at each node [43]. The term "routing algorithm" refers to a computational function that, for each packet arriving at a node, can determine the link on which the message should be transmitted. There are many routing algorithms, some of them need more information about the network in order to operate well and others need less, but each of them are more adequate for certain type of networks or other type of networks. The term "best path" could be a set of factors like: the shortest available route to destination, the fastest route to deliver the data, the most secure route, etc.

The routing algorithms of MANETs assume that there is, at least, one available path between all nodes. In fact, this is not always truth and the nodes may not be able to send data because the destination is unreachable. In these situations the data stays at the source node. However, in a DTN it is possible to deliver data in these situations. The example in Figure 2.10 shows this process [27]: the nodes A and C cannot communicate directly with each other, but they have some connectivity with node B at certain time intervals, as presented in the right graph. However, in this type of connection an e2e path is never formed between nodes A and C, and it is impossible for these two nodes to communicate in a MANET. Nevertheless in a DTN, nodes A and C may communicate because node B can take custody of the "packets" and wait for the adequate moment to deliver them.

There are routing algorithms for all type of networks, some of them are more adequate

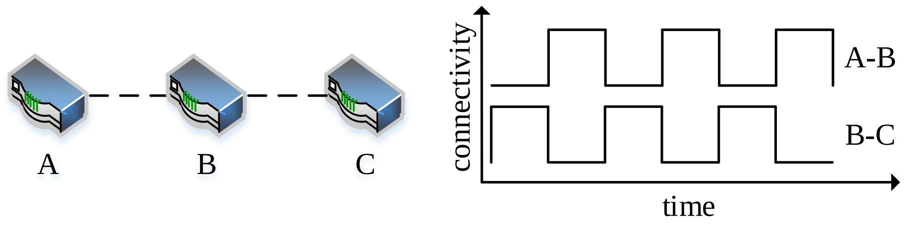


Figure 2.10: Three nodes network - based on [27]

than others. This section focuses mainly on routing in DTNs and in military networks.

2.3.2 Routing in Delay Tolerant Networks

DTNs suffer from frequent disconnections, long-duration partitioning with no e2e path. The standard routing protocols used in MANETs usually fail in this type of networks because an e2e path may be unavailable at all times, as shown in Figure 2.10. In DTNs the routing should be performed over time to achieve eventual delivery by applying persistent storage capabilities at intermediate nodes [44].

According to [45] DTN **routing presents many challenges**:

- **Contact Schedules** is one of the most important characteristics in DTNs, because it is an inter-node delay component that varies from seconds to days or even months, depending on the application under consideration. However, it is possible to classify the contacts of each type of DTNs based on the expected contact predictability, as shown in Figure 2.11.

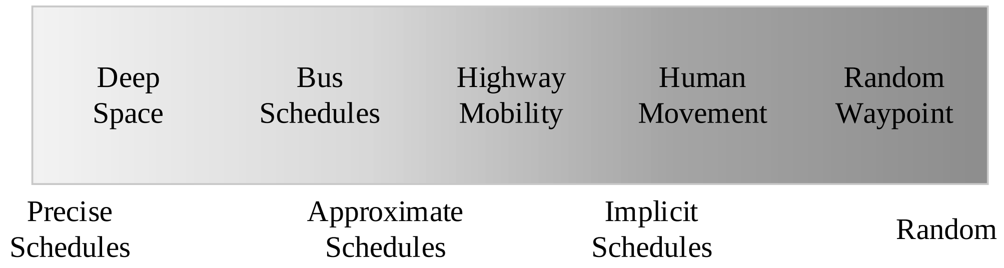


Figure 2.11: Contact Predictability Spectrum - based on [45]

At the left extreme there are precise schedules which are deterministic schedules. An example is deep space networks, where the connection windows can be calculated very accurately. One step to the right are the approximate schedules, i.e. precise schedules with low error probability. For example, a city bus VDTN where all buses have a per-established schedule, but which is not precise due to transit characteristics. However, the network behavior could be predictable. Another step to the right are implicit schedules, i.e. non-predictable schedules at time but with a considerable probability to happen. An example is a person with a free-schedule work. He goes to work all the days in the week but at each day it is impossible to determine the hour of arrival. At the right extreme are random contacts, i.e. contacts where it is not possible to predict when and if they will really happen.

  Often the DTNs are in the middle of the spectrum, because they are not completely predictable, but there are always mobility parameters that can describe, in part, these networks.

- **Contact Capacity** considers the capacity of transferring data in the available exchange time of the connection, i.e. if there is too large data and few time to transfer it, the capacity is poor. But if there is a small amount of data to transfer during contact time, the capacity is better.

- **Buffer Space** may be a challenge when the nodes experiment long periods of disconnection, because they need to keep the bundles in their memories/disks. This means that intermediate nodes should have enough buffer capabilities to support all bundles. Otherwise the routing protocols should consider the available buffer space and make better decisions taking this factor in consideration.

- **Processing Power** is a constrain in some applications where the nodes have low CPU and memory capabilities. These nodes could not be able to run complex routing protocols or maintain routing tables in their memories. Usually this is a relevant issue in WSN.

- **Energy** is a challenge in some applications where the nodes have finite energy supplies because they are mobile or they are at remote locations without access to the power grid. Routing consumes energy by performing its computational operations and by sending, receiving and storing routing messages. Thus, the routing protocol should take in consideration the energy consumption.

The DTN routing protocols may be classified according to several criteria. *Moreira et al.* performed a survey on opportunistic routing for DTNs [46] where they verified that different authors classify the routing protocols using different taxonomies. Figure 2.12 presents different taxonomies for DTN routing strategies from *Jain et al.* [44], *Zhang et al.* [47], *balasubramanian et al.* [48], *Song et al.* [49], *Nelson et al.* [50], *D'Souza et al.* [51], *Spyropoulos et al.* [52] and *Moreira et al.* [46]. For more informations consult these references.

Usually, the routing protocols implement a trade-off between controlled replication and some knowledge, because a pure-replication protocol, e.g. flooding, consumes high resources along the network, because the bundles will be sent to all nodes, and a pure knowledge protocol requires also high resources to process complex routing algorithms and

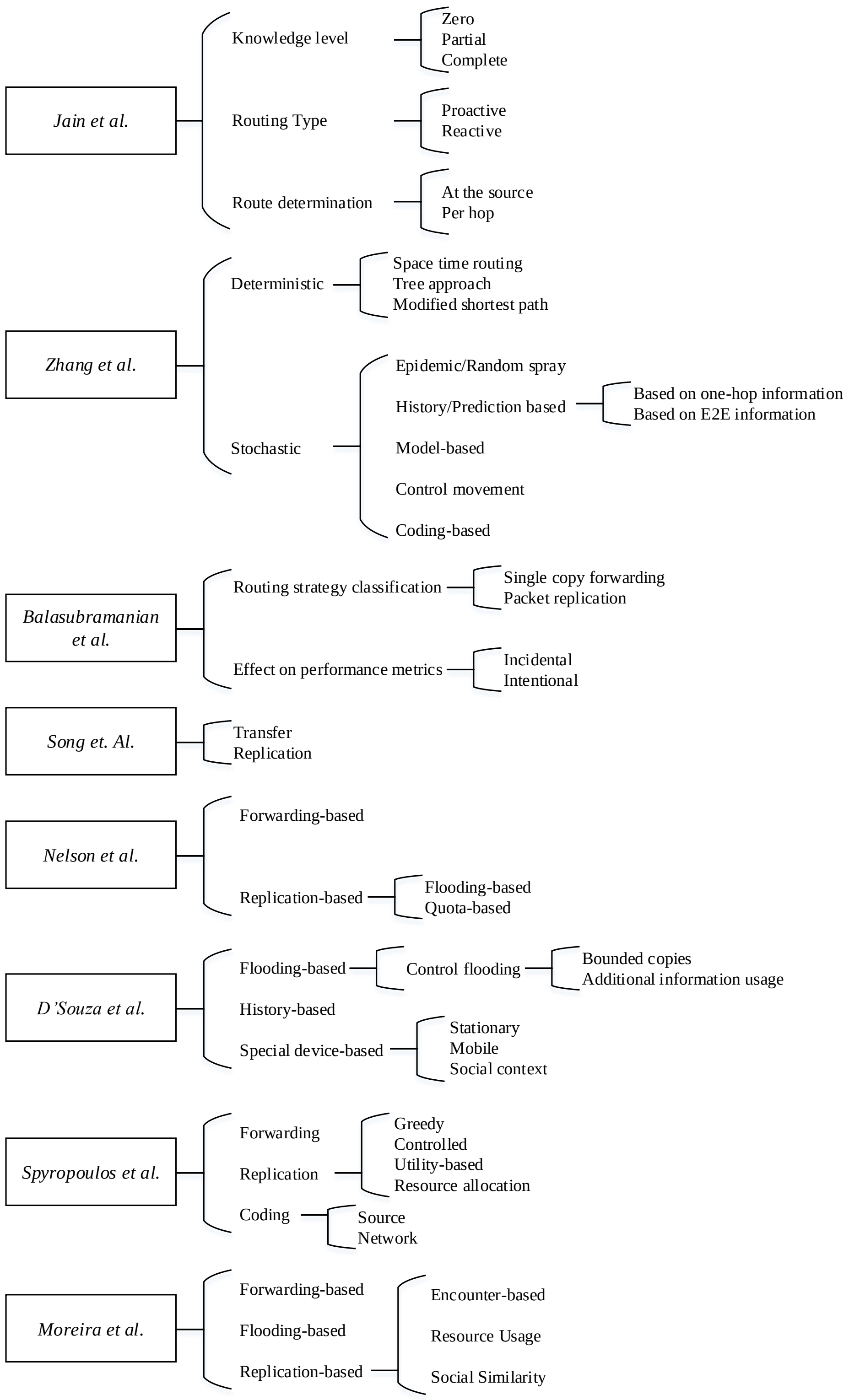


Figure 2.12: Different Routing Taxonomies - based on [46]

maintain routing tables in each node. It is necessary to find a trade-off between knowledge and replication. The trade-off is achieved most of the times by using measurements based on devices experiences to predict the movement and contacts repetition of the nodes. *Lindgren et al.* [39], *Mtibaa et al.*, [53], *Hui et al.* [54] and *Daly et al.* [55] present some works that make use of social techniques to select the best forwarders to route the information in DTNs. The routing algorithm proposed by *Lindgren et al.* is PRoPHET and it is presented in section 2.3.2.2.

The following subsections present some famous routing protocols applicable to DTNs, Epidemic, PRoPHET, Spray and Wait, MaxProp and RAPID. All these protocols operate in the bundle layer.

2.3.2.1 Epidemic

Epidemic routing [37] was originally proposed to synchronize replicated databases [56]. It is a replication-based similar to flooding routing where all nodes continuously transmit all messages to all encountered nodes that have not the messages already. The main objective is to maximize the delivery rate and to minimize the e2e delay, while also minimizing the consumed resources in the process. Considering a network of nodes with infinite storage, infinite CPU and null time to transmit messages between nodes, this routing protocol has the highest delivery ratio and the lowest delivery time. However, this assumption is not truth and all nodes continuous always transmitting the message, even if the destination received it. Consequently, this process consumes high resources along the network, mainly memory.

The basic concept of Epidemic routing is shown in Figure 2.13 with five nodes. The dark circles represents the wireless coverage of respective nodes. In Figure 2.13(a), the source node S aims to send a bundle to destination node D, but there is not an existent path between these two nodes. Thus, S transmits its message to all neighbor nodes, C_1 and C_2. Some time later, in Figure 2.13(b) nodes C_1 and C_2 move around and C_2 enters in C_3's communication range and it sends the message to C_3. As C_3 is in coverage area of D, it finally delivers the message to D.

Epidemic implements a Summary Vectors (SVs) mechanism in order to limit the message transfers. A SV contains a compact representation of the messages stored at a certain node. For example, in Figure 2.14, node A encounters B. Node A instead of transmitting all bundles, it transmits a SV that contains a compact list of its bundles (1). Node B analyzes the received SV and then transmits a vector requesting the messages that it does not contain (2). Finally, node A receives that vector and transmits only the missing bundles to node B (3).

Apparently Epidemic routing is a good routing protocol, but due the node's limited storage and limited bandwidth, it may be advisable to use other routing techniques. However, under some types of scenarios it may be the only viable option for successfully delivering data.

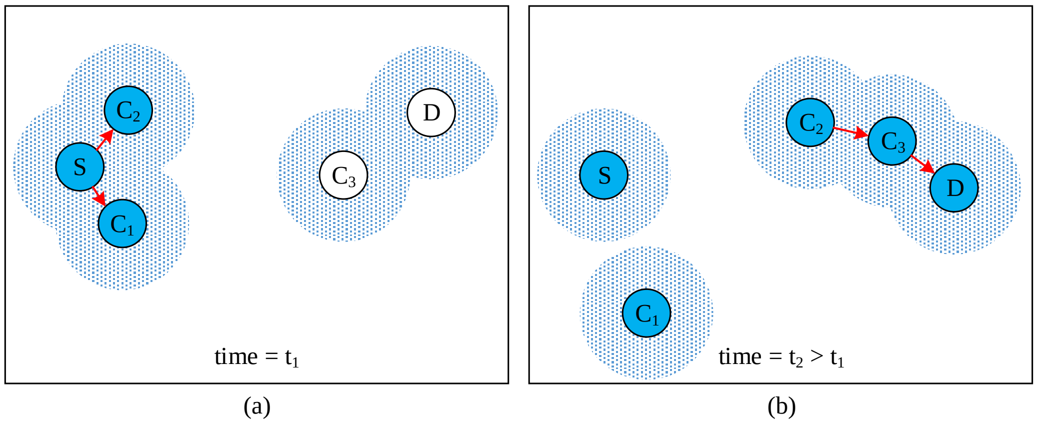


Figure 2.13: Epidemic Routing Example - based on [37]

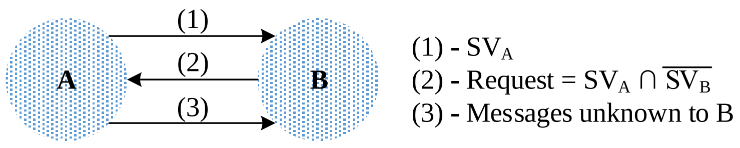


Figure 2.14: Epidemic Routing Message Exchange Example - based on [37]

2.3.2.2 PRoPHET

Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET) [39] [57] considers that the majority of the nodes do not move completely random, i.e. there are predictable movement patterns. If a location was frequently visited, it is likely that it will be visited again. This behavior is used to improve routing performance by making probabilistic routing.

PRoPHET defines a probabilistic metric called *delivery predictability* at each node for each known destination. Delivery predictability indicates the probability of a certain node to deliver a message to another node. When a node encounters another node, they exchange their delivery predictabilities map and update their local information based on that and based on the encounter. Using delivery predictability maps, the routing decisions can be made whether a node should forward or not a message to another node.

The *delivery predictability* is a Probability (P). Thus, $P \in [0, 1]$ and it should reflect the probability to deliver data to a certain node.

According to *Lindgren et al.* [39], whenever a node is encountered the delivery predictability should be updated using Equation 2.1, where $P_{(A,B)}$ is the delivery predictability from node A to node B; δ is a small positive number that effectively sets an upper bound for $P_{(A,B)}$, and $P_{encounter} \in [0, 1]$ is a scaling factor that sets the rate at which the probability increases on encounters after first encounter.

$$P_{(A,B)} = P_{(A,B)_{old}} + (1 - \delta - P_{(A,B)_{old}}) \times P_{encounter} \tag{2.1}$$

If two nodes do not encounter each other in a while, they are less likely to be good carriers of messages to each other. Therefore, the delivery predictability should reduce over the time. The Equation 2.2 allows that: $\gamma \in [0, 1[$ is the aging constant and k is the number of time units that have elapsed since the last time the metric was aged. The time units can differ and they should be defined based on the application. The delivery predictabilities are aged before being passed to other nodes. Thus, they reflect the time that has passed before they are used.

$$P_{(A,B)} = P_{(A,B)_{old}} \times \gamma^k \tag{2.2}$$

There is also a Transitive Property that is based in the following observation: if a node A frequently encounters node B and B frequently encounters node C, probably node B is a good node to forward data between nodes A and C. Equation 2.3 applies the transitive property in the delivery predictability, where $P_{(B,C)_{recv}}$ is the $P_{(B,C)}$ received from node B and $\beta \in [0, 1]$ is a scaling constant that controls the impact that transitivity has in delivery predictability.

$$P_{(A,C)} = Max(P_{(A,C)_{old}}, P_{(A,B)} \times P_{(B,C)_{recv}} \times \beta) \tag{2.3}$$

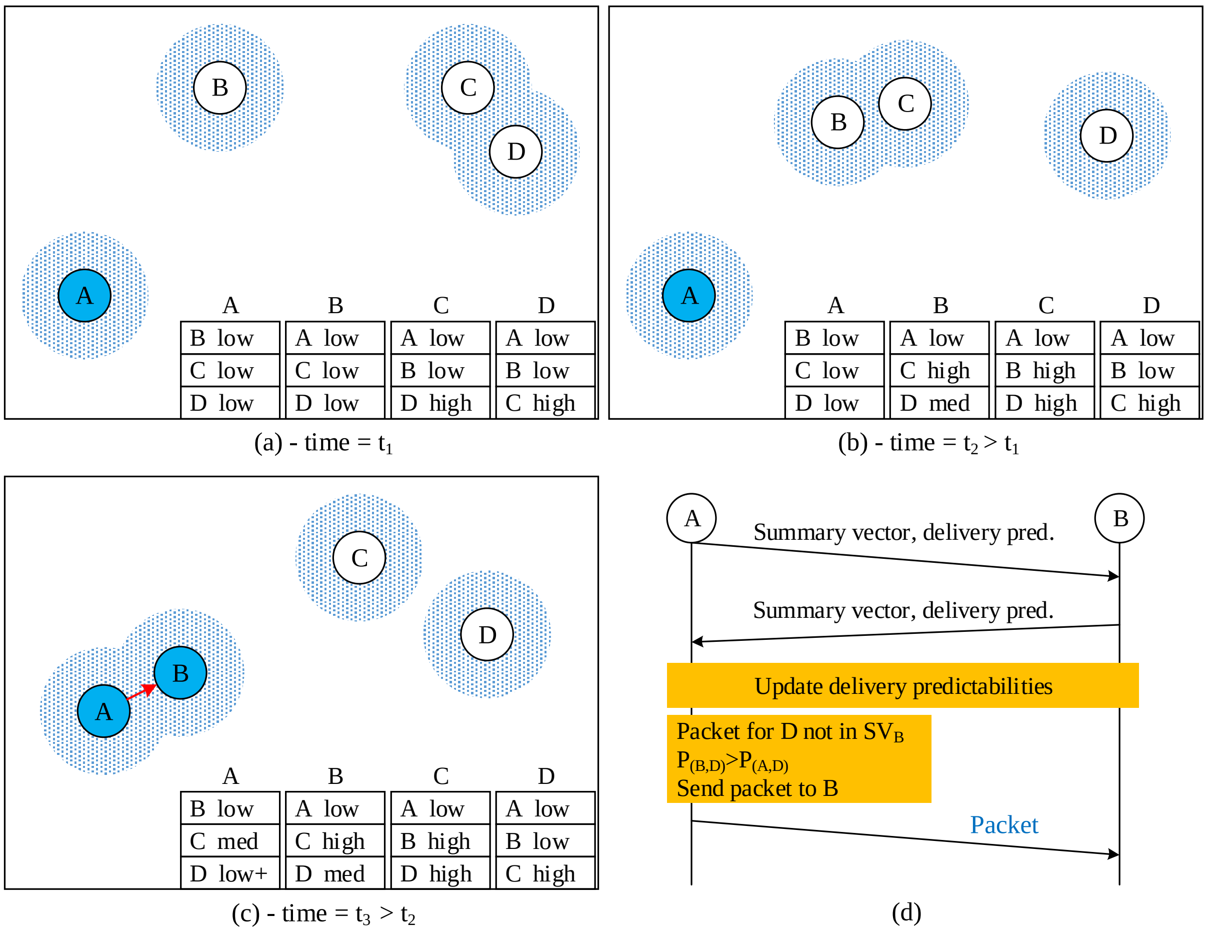
If a certain node is not present at another node's delivery predictability map, their probability is assumed to be zero.

The previous process updates the delivery predictabilities at each node to other nodes. However, there are also other factors to be taken into account. For example, the queuing policies and forwarding strategies define the rules to decide which bundles should be exchanged with other nodes at exchanging moments and these strategies affect buffer management, delivery ratio, network overhead and e2e delay [58] [39]. There are many strategies, but just two forwarding strategies were chosen to be explored under this dissertation in order to explain the basic concept. Considering that the nodes A and B are the nodes that encounter each other, node D the destination node and the bundle is presented in node A, the two strategies are the following:

- **GRTR:** Just sends the bundle to node B if $P_{(B,D)} > P_{(A,D)}$, i.e. if the other node has a higher probability to deliver it. Note that GRTR is not an acronym.
- **GTMX:** Just sends the bundle to node B if $P_{(B,D)} > P_{(A,D)} \quad and \quad NF < NF_{max}$, i.e. if the other node has a higher probability to deliver it and if the Number of Forwards (NF) at node A is lower than the limit NF_{max}. Note that GTMX is not an acronym.

Figure 2.15 presents an example to better understand PRoPHET mechanism. The example is shown in subfigures (a) to (c), wherein each subfigure presents the nodes disposition and their delivery predictability maps. Node A has a bundle to deliver to node D and the active forwarding strategy is GRTR.

At the initial time all delivery predictabilities of all nodes are low, except the predictabilities between nodes C and D because these nodes encounter themselves at that moment. Thus, they have high delivery predictabilities between them as shown in Figure 2.15(a).


B
C
D
A
A
B
C
D
Summary vector, delivery pred.
Summary vector, delivery pred.
Update delivery predictabilities
Packet for D not in SV_B
$P_{(B,D)}>P_{(A,D)}$
Send packet to B
Packet

A	B	C	D
B low	A low	A low	A low
C low	C low	B low	B low
D low	D low	D high	C high

(a) - time = t_1

A	B	C	D
B low	A low	A low	A low
C low	C high	B high	B low
D low	D med	D high	C high

(b) - time = $t_2 > t_1$

A	B	C	D
B low	A low	A low	A low
C med	C high	B high	B low
D low+	D med	D high	C high

(c) - time = $t_3 > t_2$

(d)

Figure 2.15: PRoPHET Example - based on [38]

After some time, but not much, node C moves away and encounters node B. Thus, the delivery predictabilities between these two nodes change the state to high. Besides, these two nodes trade their delivery predictability maps and due to the transitive property (Equation 2.3), the entry D in B's table changes its state to medium, as shown in 2.15(b).

Some time later, but not much, node B encounters node A. This way the delivery predictabilities between these two nodes change to high. However, these two nodes trade their delivery predictabilities maps, and due to the transitive property, node A has a medium delivery predictability to node C and has low+ delivery predictability to node D, as shown in 2.15(c). When nodes A and B trade their delivery predictability maps as shown in Figure 2.15(d), they trade also their SVs. Thus, after each node updates its predictabilities map, node A verifies that node B has a higher probability to deliver bundles to node D. Due to the fact that node A has a bundle to node D, that node B does not already have, node A decides to forward that bundle to node B and this is the essence of PRoPHET.

There are some PRoPHET variations, like PRoPHETv2 [59] proposed by the same authors with some improvements to original PRoPHET. PRoPHET+ proposed by *Huang et al.* [60] calculates a deliverability using buffer size, power, location, popularity and the predictability calculated by PRoPHET to perform routing decisions. Improved PRoPHET Routing Protocol proposed by *Han et al.* [61] is a hybrid between PRoPHET and Epidemic.
Sok et al. proposed Distance-based PRoPHET (DiPRoPHET) [62], a routing protocol similar to PRoPHET but using also the neighbors' Signal Strength Intensity (SSI) in order to fasten up the message delivery speed increasing also the delivery ratio. It considers a small value, function of distance, in order to increase or decrease a bit the probability values obtained from PRoPHET. However, this work performs its implementation in one oldest version from PRoPHET presented in [57] which in turn, it had several modifications and improvements that are updated in [39].

2.3.2.3 Spray and Wait

Spray and Wait [63] is a simple routing protocol, yet efficient, that attempts to gain delivery ratio by limiting the number of copies per message allowed in the network. The Spray and Wait operation can be divided in two phases:

- **Spray phase:** When a new message is originated at source node, just L message copies can be initially spread/forwarded by the source or other nodes that received a copy (depending on spray strategy), i.e. the spray phase happens when the nodes are allowed to transmit the message.

- **Wait phase:** if the destination is not found in the spray phase, each one of the L nodes carrying the message is just allowed to transmit the message only to the final destination.

There are different forms to process the spray phase [14]. The most common ones are:

- **Vanilla** is the simplest way to apply spray phase. Just the source node can spray messages, i.e. source node can send up to L copies of the message to L neighbors. Each of one these neighbors just send the message to the final destination.

- **Binary:** The source node starts with L copies of the message that it is allowed to send. When the source encounters another node, it sends to the new node $\lfloor L/2 \rfloor$

copies of the message and retains $\lceil L/2 \rceil$. Each node that has $n > 1$ copies of the message and encounters another node without copies, sends to it $\lfloor n/2 \rfloor$ and keeps $\lceil n/2 \rceil$ copies. When a certain node eventually gives away all of its copies, except for one, it enters in the wait phase. The benefit of this form is that messages are disseminated faster than in vanilla.

2.3.2.4 MaxProp

MaxProp [64] is a flooding-based protocol, which may be considered replication-based, due to the fact that it aims to make a better use of the bandwidth and buffer space to unify the problem of discarding and scheduling transmissions of the messages. MaxProp evaluates a cost of virtual e2e path to reach the destination using an estimation of the route failure probability. At the beginning, the failure probability of neighbor nodes is uniformly distributed for the network. As time goes by, it is updated using an incremental average. In addition to that, the message freshness if evaluated using its average transferred size. If the number of hops of a specific metric is lower than its freshness, the message is classified according with its number of hops; otherwise, it is prioritized using the previous mentioned cost. Moreover, this protocol informs its neighbors to clear the possibility of existing copies of the delivered messages using broadcast acknowledgments through all the network.

2.3.2.5 RAPID

The Resource Allocation Protocol for Intentional DTN (RAPID) [48] is a replication-based routing protocol which aims to optimize a specified routing metric. To achieve that, it uses a random variable to represent the encounter between two nodes, and propagates those messages through the network. The dissemination of these messages are according to their marginal utility. The marginal utility is calculated using an utility function which is based on a ratio between message size and the decreased delivery delay associated with it. After that, only the messages with a positive marginal utility are replicated in the network. Thus, this protocol is replication-based where the replication is controlled by the marginal utility. The metrics that this protocol may optimize using the utility function are: average delay, missed deadlines, and maximum delay.

2.3.2.6 Comparison between DTN routing protocols

Table 2.2 presents a high level comparison between the DTN routing protocols presented in section 2.3.2. The comparison focuses in the main characteristic of each protocol presenting their advantages and disadvantages.

These protocols are all based on the replication of bundles. However, they have differences mostly in the criteria that is used to perform the decision, i.e. when a bundle should be replicated or not. However, the majority of these criteria are to control the replication based on several metrics to minimize the resources consumption or the expected delay (they do not use social metrics). However, PRoPHET controls the replication (and the resources consumption in the network), while it targets the message to the destination,

Table 2.2: Comparison between presented DTN routing protocols

Routing Protocol	**Forwarding/Replication type**	**Advantages**	**Disadvantages**
Epidemic [37]	Based on pure replication.	Ease to implement and simple routing logic.	No knowledge about the network. Decision depends only if the other nodes have the bundle. Consumes high resources along the network.
PRoPHET [39] [38] [57]	Based on controlled replication by using a probability based on the historical of encounters (social-based).	The number of replicas in the network is more limited based on the neighbors' probability to deliver the message. Consequently this protocol does not consume as many resources as Epidemic.	Complex to implement. It is necessary to use resources to calculate the probability and to consume memory to maintain the predictabilities table at each node, mainly in dense networks.
Spray and Wait [63]	Based on controlled replication by limiting the number of copies per message.	Ease to implement. The number of replicas of the message in the network is controlled by the source node that limits the number of copies allowed.	No knowledge about the network. Decision depends only of the number of remaining bundles.
MaxProp [64]	Based on Flooding/Replication.	It maintains a sorted list of bundles. This sort is based in the estimated probability of that a transitive path will be formed, which will the path to replicate the packets.	It overloads the network when the routing tables are exchanged. It is not adequate to disperse networks, because the nodes cannot maintain the graphs of the connected nodes.
RAPID [48]	Based on Flooding/Replication.	It allows to control the resource consumption by controlling the replication of bundles.	It may overload the resources when the nodes exchange their meta-data.

because it uses social-based metrics, where each node maintains predictability tables to each node. Thus, this routing protocol replicates the message to the neighbors that have higher probability to deliver it. This protocol is then the chosen one to be the basis of this work.

2.3.3 Routing in Military Networks

Military communications at terrain are increasingly network-based and must be considered communication networks. These networks are becoming less infrastructured because at certain scenarios it does not exist an infrastructure to support communications, or the existent one crashed due a catastrophic disaster, for example.

Figure 2.16 exemplifies that military networks may operate in challenged environments with many constrains, e.g. intermittent connectivity, high mobility, poor channel quality, low bandwidths and/or network fragmentation. The Army Communications-Electronics Command (CECOM), Defense Advanced Research Projects Agency (DARPA), commercial industry among others believe that, one of the main keys to increase the military network's capacity[2] is to use more efficient routing protocols [65], which is one of the main focus of this dissertation.

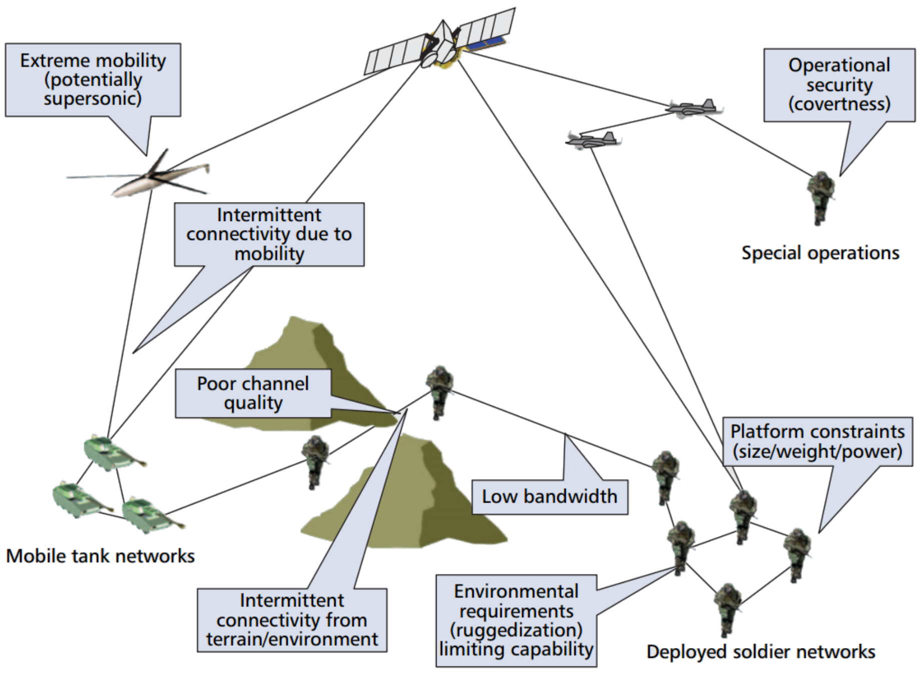

Extreme mobility (potentially supersonic)
Operational security (covertness)
Special operations
Intermittent connectivity due to mobility
Poor channel quality
Platform constraints (size/weight/power)
Low bandwidth
Mobile tank networks
Intermittent connectivity from terrain/environment
Environmental requirements (ruggedization) limiting capability
Deployed soldier networks

Figure 2.16: Constrains of Military Environment - adapted from [66]

Tactical military networks, at network-centric environments, are mobile and often Ad-Hoc, due to the fact that these networks are self-configuring without any central controller

[2]Capacity defines the amount of data that a network can functionally support.

entity. Thus, it is necessary to use routing protocols to address data in these networks, allowing a multi-hop communication, but the majority of the commercial protocols may be not adequate in tactical and emergency environments or congested networks due to their constrains. It is necessary to develop new protocols specifically for tactical forces, and it is expected that these routing schemes can use the network more efficiently, by reducing redundancy and excess overhead while take advantage of knowledge of the network state to improve themselves.

The following subsections present Terrestrial Trunked Radio (TETRA) (a proprietary solution for communications), Military Networks as MANETs and Military Networks as DTNs.

2.3.3.1 Terrestrial Trunked Radio (TETRA)

Terrestrial Trunked Radio (TETRA) [67] is a set of standards developed by European Telecommunications Standards Institute (ETSI) to digital trunked mobile radios to satisfy the needs of organizations such as: Military, Government, Utilities and others. TETRA supports voice, data transfer and supplementary services/applications, like group and private voice services, call forward and others [68].

Many Professional Mobile Radio (PMR) communication systems rely on TETRA standards for providing mission-critical communication environments. TETRA, as an ETSI approved system, has a commercial advantage within Europe, both on the point of view of manufacturers, operators and governments which in Europe uses ETSI approved systems for their contracts. However, TETRA has a level of complexity that turns the infrastructure and the terminals expensive [69] and it works mainly in direct communications. However, it is also possible to use an intermediate node as a relay point.

2.3.3.2 Military Networks as MANETs

According to *Burbank et al.* [66], it is expectable that future tactical actions will achieve an Internet-like capability providing network access to offer communication everytime and everywhere. The idea to achieve this considers a Global Information Grid (GIG) that includes different networks, ground-based wired and wireless, satellite, shipboard, airborne, etc., as shown in Figure 2.17. *Burbank* believes that these networks will be connected via IP networks.

In an ideal MANET, if all nodes are directly or indirectly connected, i.e. without network fragmentation, the structure presented in Figure 2.17 represents a high-level view of their functionalities. There is the presence of a fixed terrestrial network where the routing protocols are actually well developed. There is also several MANETs that may include hundreds or thousands of mobile nodes which can connect to the infrastructured network.

However, the concept of ideal MANET does not exist in military scenarios. The high quantity of nodes and the environment constrains affect the routing protocols efficiency,

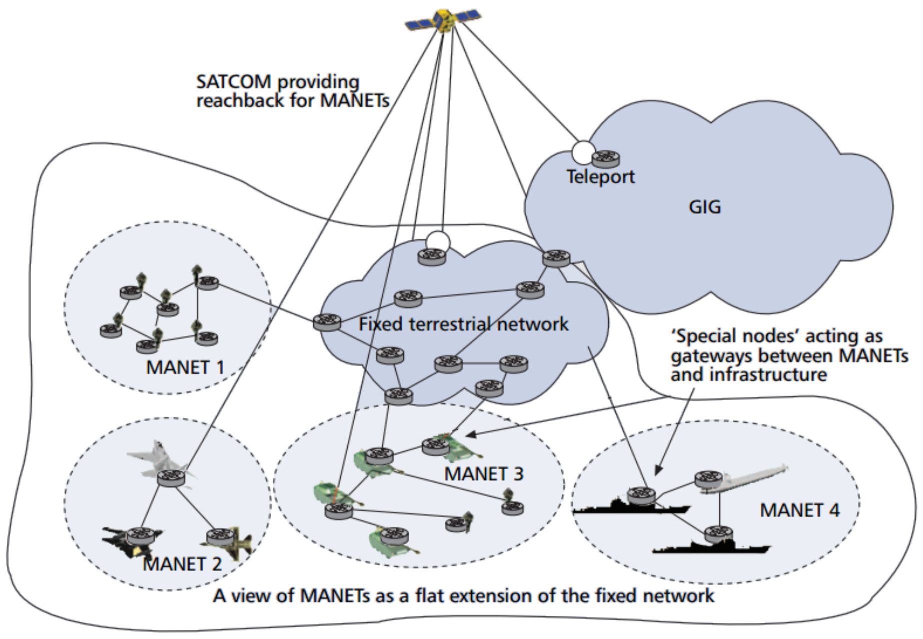

SATCOM providing reachback for MANETs
Teleport
GIG
Fixed terrestrial network
'Special nodes' acting as gateways between MANETs and infrastructure
MANET 1
MANET 2
MANET 3
MANET 4
A view of MANETs as a flat extension of the fixed network

Figure 2.17: Military MANET as extension of network infrastructure - adapted from [66]

specifically the high mobility and high intermittency (Figure 2.16) that trigger many route updates, which leads to high routing overhead, high computation and even the protocols may diverge. At the best of my knowledge, actual MANET routing protocols do not work well with scalable networks and intermittent connections [70] [71], even if the mobility in military network presents some coherence and the nodes are more concentrated [66].

2.3.3.3 Military Networks as DTNs

To lead with the disruptions, interference, network partitions and the other constrains presented in Figure 2.16 some authors suggest DTNs as a solution to deal with them. ”*DTN technology is one of the key technologies to solve these problems. At present, DTN technology has been paid attention to various fields around the world*” - *Lu et al.* [11].

Parikh et al. [72] present an overview relative to apply DTNs technology in the United States of America (USA) Marine Corps C2 On-the-move Network Digital Over-the-horizon Relay (CONDOR). It was built a prototype where an adapted version from DTN2 was installed in a Cisco Router. These authors conclude that DTN ”*seeks to deal with the realities of military tactical communication*”. According to *Jonson et al.* [73], DTNs are applicable to airborne networks characterized by highly intermittent links and long link delays. *Rigano et al.* [74] declare that DTNs provide a robust alternative to only-IP models in environments characterized by long delays or temporary network partitions, and achieve equal performance in environments where only-IP models are feasible. It is the concept of store-and-forward that allows to deal with these constrains, but it is also necessary an adequate routing protocol to work under these circumstances.

Disruption and interference are characteristics of the Physical environment, i.e. they depend of the media conditions; the current routing protocols for DTNs, presented in section 2.3, do not consider PHY parameters to try to deal with these constraints. At the best of my knowledge, there are no routing protocols to DTNs that consider the links conditions, i.e. the quality of the links.

There are also few works that apply DTNs in military networks. There some theoretical studies and references about this subject. For example, *Fall et al.* [7] suggests to use DTNs in military scenarios but only as an example to the applications of DTNs. *Jonson et al.* [73] presents the application of DTNs in airborne networks, but is also a theoretical work that presents some use cases and challenges. *Rigano et al.* [74] presents another theoretical work about the subject, and it describes an approach to improve network performance in naval networks using DTNs with a transport protocol called NACK Oriented Reliable Multicast (NORM). *Lu et al.* [11] presents also a study about DTN applications in military communications. In practical terms, the work from *Parikh et al.* [72], presented in this section, is the most practical case that I found about to apply DTNs in military networks.

2.3.3.4 Comparison between the presented military networks

Table 2.3 presents a high level comparison between the military networks presented in section 2.3.3. The comparison focuses in the main characteristicss, presenting their advantages and disadvantages.

The DTNs are chosen as the privileged network to be used in this work, because as previously stated, they have store, carry and forward mechanisms, which enable the nodes to store the packets, when they cannot deliver them to other nodes. This is the main advantage of these networks, when compared to the Ad-Hoc Networks, which allow to improve the routing in the case of network fragmentation. Besides, the DTNs (and Ad-Hoc networks) may be implemented in off-the-shelf devices and they allow to insert new functionalities very easily, unlike TETRA, which is a proprietary and expensive solution. Besides, it is mainly infrastructured, with some Ad-Hoc capabilities.

2.4 IEEE 802.11 WLAN

2.4.1 Overview

The Institute of Electrical and Electronics Engineers (IEEE) Standard 802.11 [75] describes the functional components of IEEE 802.11 Local Area Networks (LANs). It is important to introduce it due to the fact that, in this dissertation, the MAC and PHY layer functionally defined in the Standard are used to obtain metrics that allow to represent the quality of the wireless links. The location of the IEEE 802.11 Standard in the OSI model is presented in Figure 2.18. It describes the MAC (from Data Link Layer) and PHY specifications which are significantly different from traditional wired LANs. It only describes the functional components, instead of the physical implementation details.

Table 2.3: Comparison between the presented military networks

Military Network Type	**Advantages**	**Disadvantages**
TETRA [67]	It uses specialized hardware prepared to military scenarios. Promotes a standard accredited by ETSI which different forces may use with confidence. It may be integrated with telephony network.	It is expensive. It does not support multi-hop wireless communications with more than 1 hop.
Ad-Hoc	It supports multi-hop communications. There are lots of routing protocols developed to Ad-Hoc networks. It supports integration with with IP networks. It may be applied in off-the-shelf devices. It is ease to insert new services.	The best performance is achieved only with the network formed, i.e. without fragmentation. There are few quality-based routing protocols developed.
DTN	It supports store-and-forward. It supports integration with IP networks. There are lots of routing protocols developed to Ad-Hoc networks. It may be applied in off-the-shelf devices. It is ease to insert new services.	There are few works about to apply DTNs in military scenarios. There are few quality-based routing protocols developed.

2.4.2 Architecture

The IEEE 802.11 architecture offers several components that interact to provide a WLAN where the Stations (STAs) may move transparently to upper layers.

IEEE 802.11 is based on the cell architecture. When each of these cells is controlled by an AP these cells are called Basic Service Set (BSS) (infra-structured mode), but when there are not APs, these cells are then called Independent Basic Service Set (IBSS) (Ad-Hoc mode). A set of BSSs that are interconnected between them are called Extended Service Set (ESS) and the Domain System (DS) is the communication way between BSS's APs, which is usually the Ethernet. To identify these cells, the most common way is the Service Set Identifier (SSID) (32 bytes String) that assigns a name to a particular IEEE 802.11 WLAN. Each ESS has to maintain a unique SSID to its BSSs. A visual representation is

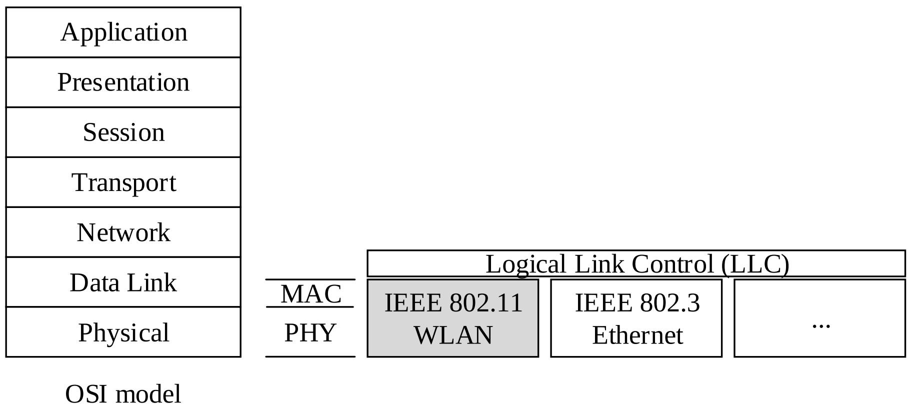


Figure 2.18: IEEE 802.11 and OSI Model - based on [76]

presented in Figure 2.19.

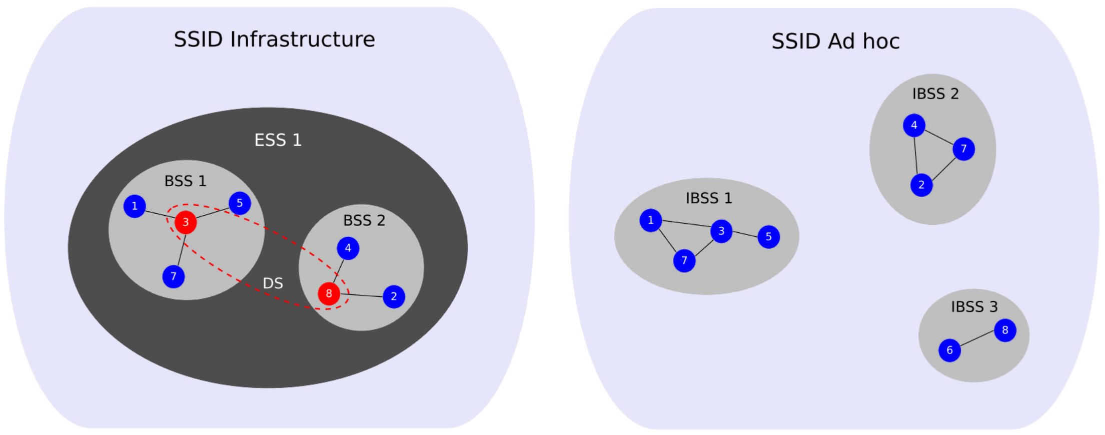


Figure 2.19: High Level IEEE 802.11 Architecture [77]

2.4.2.1 Infrastructure vs Ad-Hoc Operation Modes

The main difference between Infrastructure and Ad-Hoc networks is the absence of an central entity, the AP, in Ad-Hoc mode as shown in Figure 2.20. In the infrastructured mode there is a presence of one, or more, APs and the STAs (nodes) can just communicate with reachable APs. In Ad-Hoc mode only the STAs exist, and these can communicate directly with other reachable nodes. Multi-hop communications are also possible if an adequate routing protocol is running.

In this work it will be used the Ad-Hoc communication mode to allow several mobile nodes to emulate military nodes, such as different moving units in a military team.

In both modes of operation, if a STA aims to communicate with the AP or another node, it needs to associate itself first with the AP's BSS or other node's IBSS, respectively.

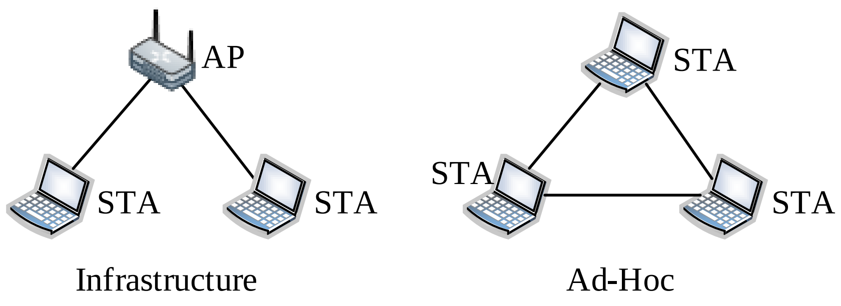


Figure 2.20: Infrastructure vs Ad-Hoc

There are two ways of doing this, either by using Passive Scanning/Probing or by using Active Scanning/Probing. The difference between these two modes is that, in Passive Scaning the AP in Infrastructure mode or the nodes in Ad-Hoc mode are regularly sending beacons which contain a Sequence Number (SN) that will be used in this work.

2.4.3 Media Access Control (MAC) Layer

The Media Access Control (MAC) Layer is the lower part of the Data Link Layer from OSI model as shown in Figure 2.19. The MAC layer offers addressing and channel access control mechanisms that enable several devices to communicate in a specified network. This layer specifies several modes of media access, like Distributed coordination function (DCF), Point Coordination Function (PCF), Hybrid Coordination Function (HCF) and others that are defined in Chapter 9 (MAC sublayer functional description) of IEEE 802.11 Standard [75].

The IEEE 802.11 MAC also defines the format of several frames (packets) to perform certain tasks or transfer data. Some of these frames contain informations that may be useful to evaluate the quality of the wireless links, and this section presents the way to reach the SN, which is one of the metrics that will be suggested in this work to compose the metric to evaluate the quality of the links.

The following descriptions related to frame format are necessary because of their importance for the work developed in this dissertation.

The MAC frame format is shown in Figure 2.21. It comprises of a set of fields that appear in a defined order in all frames. Frame Control, Duration/ID, Address 1 and Frame Check Sequence (FCS) corresponds to the minimal frame format that need to be present in all frames, including reserved formats. Address 2, Address 3, Sequence Control, Address 4, Quality of Service (QoS) Control, High Throughput (HT) Control (only for IEEE 802.11n) and Frame Body are present only in certain frames.

A detailed description about the frame format specifications is presented in Chapter 8 (Frame Formats) of the IEEE 802.11 Standard [75].

The Frame Control field from MAC frame is presented in more detail in Figure 2.22.

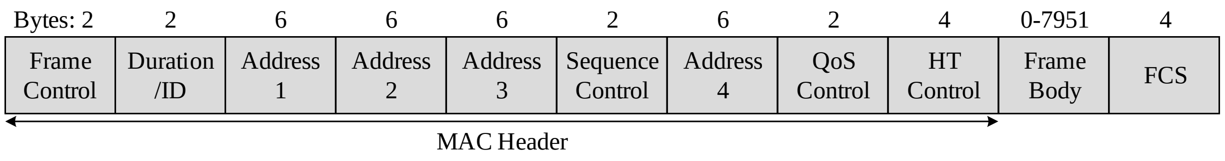


Figure 2.21: MAC Frame Format - based on [75]

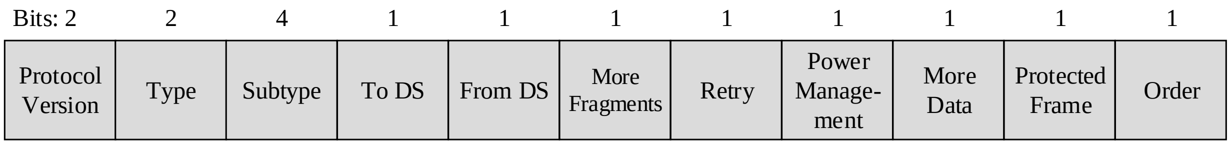


Figure 2.22: Frame Control Field - based on [75]

The type field defines the type of the frame. Available types are Management, Control, Data Exchange and some that are reserved for future purposes, that are not yet completely defined. The subtype field defines the subtype in the context with the type field. For example, some subtypes of management frames are:

- **Beacon** is generated periodically with the purpose of announcing the presence of a STA and contains several informations, e.g. Timestamp, Beacon Interval, SSID, etc.
- **Association Request** is used when a STA aims to associate to a BSS or an IBSS.
- **Association Response** is the answer to an Association Request that can be positive or negative. In case of a positive answer, the frame includes information regarding the association, e.g. supported data rates.
- **Reassociation Request** is used when a STA looses signal from one AP and finds another AP It sends a reassociation request to the new AP. This may trigger the forwarding of data queued in previous AP to the STA.
- **Reassociation Response** is the answer to a Reassociation Request.
- **Probe Request** is used in Active Scanning mode, when a STA needs to obtain information about the neighboring STAs.
- **Probe Response** is the answer to the Probe Request with the information about the STA.
- **Disassociation** is used when a STA aims to finish the association with an AP that is previously established. Thus, the AP can free some resources relative with the presence of this STA in its local tables.

- **Deauthentication** is used when a STA decides to finish the communication with the AP. For example, when too many beacons are lost.

The Sequence Control presented in Figure 2.23 is a two byte field contained in the MAC Frame (Figure 2.21). The sequence control has 16 bits with two subfields, Sequence Number (SN) and Fragment Number. The Sequence Control Field is not present in Control Frames.

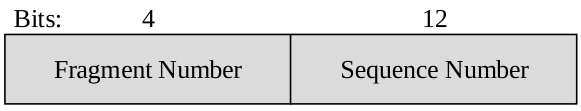


Figure 2.23: Sequence Control Field - based on [75]

The SN subfield has 12 bits and it indicates the sequence number of the frame by frame's type. Each frame transmitted by a STA has a sequential SN, except the Control Frames. Beacons are one example where the SNs exists. This SN is the metric that will allow to evaluate, in part, the quality of the links. The Fragment Number subfield has 4 bits and indicates the number of the fragment of the frame, if they are fragmented. Otherwise, the Fragment Number is '0'.

2.4.4 Linux Wireless Subsystem

The Linux Wireless Subsystem [78] contains several modules to deal with IEEE 802.11 hardware configurations and to manage the packets transmission and reception [79]. Figure 2.24 presents an abstracted view of the interaction between different layers of the Linux Wireless Subsystem. cfg80211 [80], mac80211 [81] and drivers are inside kernel domain and Applications for control/management are contained in User Space. Applications can use the Netlink interface nl80211 [82] to access the cfg80211 module.

The cfg80211 module is the Linux configuration API. It contains the common IEEE 802.11 configuration options and performs active tasks like scanning for APs, managing security keys and virtual interface management.

The mac80211 implements in software the MAC layer, i.e. it is a softMAC device driver. If the implementation would be made in hardware, it would be a fullMAC. Its configurations are handled by cfg80211 (through nl80211) or wireless-extentions (wext) due to legacy reasons. mac80211 interacts directly with the IEEE 802.11 device drivers, like ath5k, to transmit or receive frames. It supports IEEE 802.11a/b/d/g/n/s, QoS mechanisms and different types of interfaces, like AP, IBSS, monitor[3] and others. mac80211 has also support to multiple virtual interfaces (wifi-iface) using the same radio interface (wifi-device). For

[3]Monitor mode is usually a passive mode, where no packets are transmitted and the interface does not join to any network. The interface receives all packets from the driver, even if they are not destined to it. mac80211 module sends upstream all packets, including extra header information if it exists.

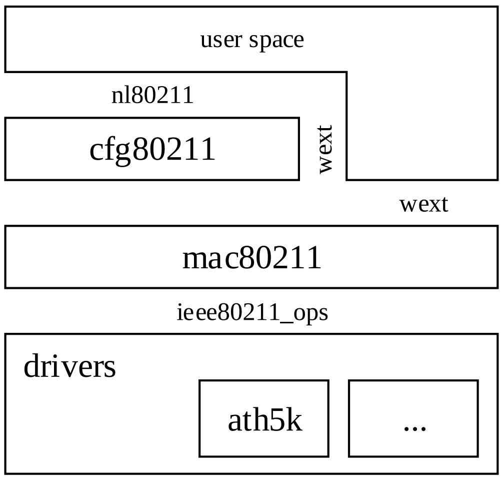


Figure 2.24: Linux Wireless Subsystem - based on [79]

the OS, Virtual Interfaces are seen as other radio interfaces, and the OS as well as its applications can use both simultaneously [83].

Nowadays, there is an extensive list of working Linux wireless drivers that support most of the available networking cards, although the majority of drivers do not implement all the features. A more detailed description of these cards and features is presented in [84].

A few years ago, Linux started adding the possibility to use Radiotap [85] on its drivers and APIs. mac80211 is able to receive Radiotap Headers before the IEEE 802.11 general frames. When a driver adds a Radiotap, it enables a specific flag to inform the mac80211 module. When mac80211 module aims to receive Radiotap Headers, it can signal the driver. This happens when mac80211 is configured with a monitor interface.

Radiotap [85] is a *de facto standard*[4] mechanism that provides additional information about frames.

A Radiotap capture begins with a Radiotap Header shown in Figure 2.25.

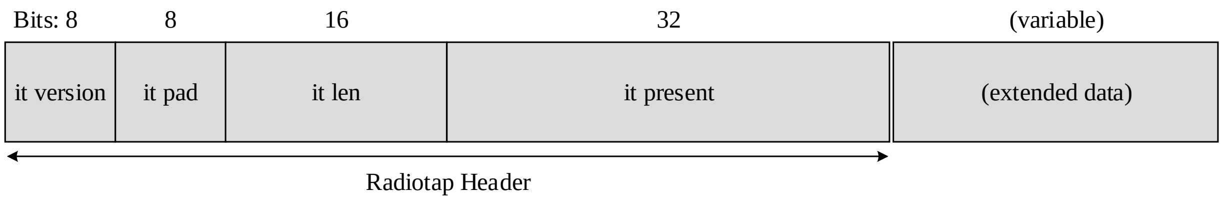


Figure 2.25: Radiotap Structure

[4]A *de facto standard* is a formal or informal standard that achieved a dominant position by tradition or market dominance. There might not exist official standardized documents.

The *it version* (8 bits) indicates the version of radiotap header in use. Actually there exists only the version 0. The *it pad* (8 bits) is currently not used, it exists for alignment purposes. The *it len* (16 bits) contains the total length of radiotap data, including the header. This is useful for a developer/programmer to know where is the beginning of IEEE 802.11 general frame if its radiotap parser does not know how to understand all presented data fields. The *it present* (32 bits) is a bitmap that indicates which data fields are present after the Radiotap Header. If the 31st bit of this field is not set, that means that there is radiotap data after the header and this data is strictly ordered according to *it present* bits. These data might contain several informations, such as physical information. For example, the packet's **SSI**, the timestamp, the modulation scheme and/or others.

2.4.5 Gather MAC and Physical Layer Information from Application Layer

During the research about methods to gather IEEE 802.11 MAC and Physical information from Linux Wireless Subsystem it was verified, at the best of my knowledge, that there are essentially two different options for an Application to gather information from the Linux Wireless Subsystem. The first option consists of using the nl80211 interface (Figure 2.24) and gathering the **connection status** information. The second and more complex option consists in using a monitor interface to **capture wireless frames** and extract the relevant information [79]. These two options are explained in the following:

2.4.5.1 Gather Connection Status

This option uses the nl80211 interface which describes, very straightforwardly, the commands that applications may use and which information can be retrieved from cfg80211.

The iw command [86], the new version of deprecated iwconfig [87], is an application that interacts with nl80211 to offer to the user a command-line interface for wireless devices for configuration and connection status visualization. However, this option is not the best choice for demanding applications that need more information, because the information that can be retrieved from nl80211 is not sufficient and there are some open issues, like the reported values are generally averaged, the method how the values were generated is not documented and the updating frequency is unknown. Furthermore, if an application needs to know a certain information regularly, it needs to perform polling of the nl80211 interface which may be computationally demanding [79].

Due to these issues, the nl80211 interface was not used in this work because there is a need to regularly obtain precise information which is not plausible due to the limitations stated before.

2.4.5.2 Capture Wireless Frames

This different option consists in capturing all frames, in the media, and filtering them, in order to gather and process relevant data, and it allows to obtain information relative to each received packet/frame.

Usually a monitor interface is used to capture all frames (at the same physical channel), including the Radiotap Header that includes extra information related to the frame, but configuring a monitor interface does not allow association to any network, which might be an obstacle to the applications. However, mac80211 allows virtual interfaces as explained in section 2.4.4, and it is possible to capture all network packets while concurrently using the wireless card for user's operations.

There are some well known applications that are able to capture packets and present them to the user. For example tcpdump [88], a command-line packet analyzer (lightweight for embedded systems) or wireshark [89], the world's foremost network graphical protocol analyzer capable of capturing packets, parsing and many other features. However, these applications only present the handled data for the user, and it is difficult or not sufficient to integrate them with other applications.

Those well-known applications and eventually other user's applications use pcap, an API for **p**acket **cap**ture offered by libpcap library [90]. pcap may capture packets from various networks, such as wireless 802.11g, ethernet and others, and it has also the ability to filter packets based on Berkeley Packet Filtering (BPF) [91], implemented directly in OS kernel for better performance [92]. The pcap capture process is presented in Figure 2.26. When the card driver receives a packet, it sends the packet to the protocol stack as usually, but it sends also the packet to the BPF (Packet Filter) that is responsible to filter packets and send them to the subscriber applications [93].

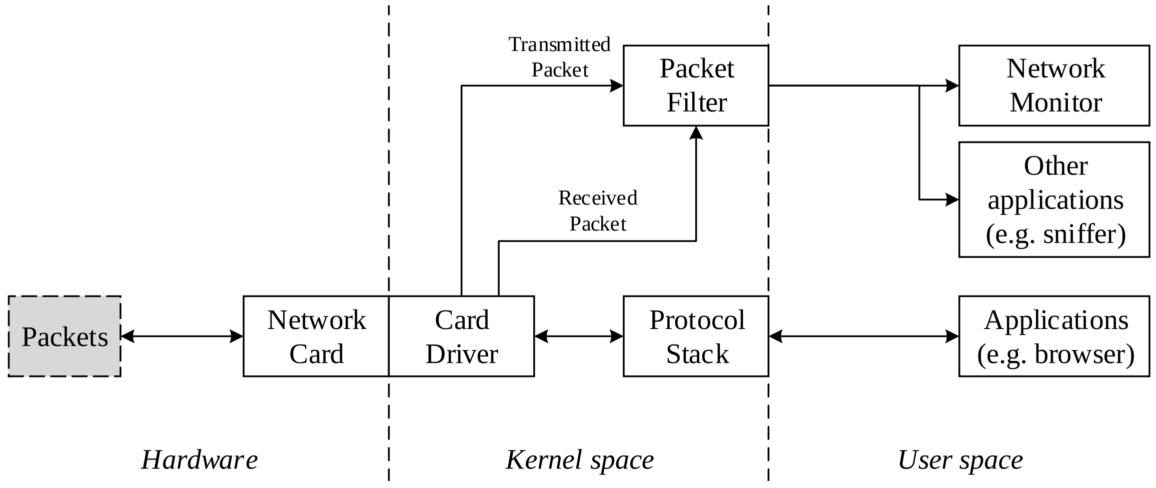


Figure 2.26: Pcap capture process - based on [93]

In the work presented in this dissertation, it is chosen the method of capturing wireless frames because it offers more flexibility, more confidence and it is the method used by the well-known applications, e.g. tcpdump.

2.5 Chapter Considerations

This chapter provided a study about three relevant subjects to this dissertation, DTNs, Routing and the IEEE 802.11 WLAN standard.

First, the state of the art about DTNs was presented which are one of the main topics of this dissertation. In this section it was presented the DTN definition, the applications, the architecture and bundle protocol, and also several implementations in order to give the reader the necessary knowledge about DTNs.

Then, the state of the art on DTN routing was presented. Then, it presented routing in military networks, including some studies about the applicability of DTN routing in military scenarios.

Finally, a brief resume of IEEE 802.11 WLAN standard was presented as well as the implementation of the standard in Linux systems. Also, some techniques were presented to gather and capture certain information about the received packets in the wireless drivers.

These topics are the basis to understand the work developed; the work in the following chapters contains several references to this chapter, due to the relevance of these subjects to perform the quality-based routing work.

Chapter 3

Scenarios and Proposed Solution

3.1 Introduction

This chapter presents the navy scenarios for communication presenting the nodes topology and the main constrains. These scenarios will be the reference to the scenarios used to evaluate the quality-based approach in section 5.6. Then, it presents the proposed solution for quality-based communications in these scenarios.

Section 3.2 presents the three navy scenarios for communication: Inspection/Boarding, Naval and Amphibious Operations and the Population Support Scenario, with their characteristics, communication and routing challenges.

Section 3.3 presents the quality-based DTN to allow communications in the aforementioned scenarios. First, it presents the definition of link quality which is important to define and compare the quality between different links. It presents also the Q-PRoPHET, a DTN routing protocol that performs decisions based on the quality of the links. This solution is addressed to solve the navy's communication challenges presented in section 3.2.

Section 3.4 presents the chapter considerations.

3.2 Navy Scenarios

The navy scenarios presented in this section are derived from practical actions observed during navy tactical actions where the need to have communication between different elements is important. The scenarios are as follows.

3.2.1 Inspection/Boarding

The Inspection/Boarding Scenario occurs at the sea and selects one team to inspect a Foreign Boat (FB). This team needs to maintain communications with the Mother Ship (MS), either directly or via an Intermediate Boat (IB) if necessary, as shown in Figure 3.1.

The scenario is characterized by low mobility, where the objective is to optimize the coverage area without any obstacles in sight, but possibly with some interference in the media due to reflections in the water, intentional jamming, environment factors, etc. These factors may affect the communication between the Mother Ship and teams in the Foreign Boat, whether it turns the communication intermittent or even disconnects both nodes. Therefore, it is possible or even necessary to use an Intermediate Boat as a communication relay in order to allow communications between the Mother Ship and Foreign Boat.

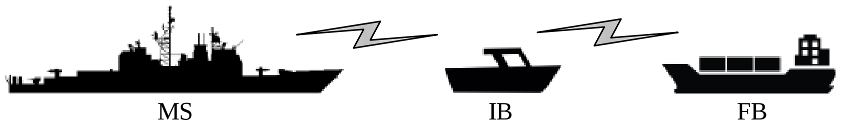


Figure 3.1: Inspection/Boarding Scenario

It is necessary that the Mother Ship automatically knows that it has two neighbors, and that it is able to communicate with both, choosing the best path(s), either directly or multi-hop, depending on the situation.

3.2.2 Naval and Amphibious Operations

The Naval and Amphibious Scenario occurs at the sea and/or at shore, and the objective is to offer communication between the Commanders (C) and the Teams (T) in the boats or in the shore as shown in Figure 3.2. It may be necessary a presence of Helicopters (H) or other mobile nodes to carry data. These scenarios are characterized by the nodes presenting high mobility and it is expected failures in the communication due to interference or partition of node groups. In this scenario there should exist communication between all elements, and the intermediate nodes may have to forward data between Commanders and Teams because these two may be directly unreachable.

This scenario is characterized by the nodes presenting mobility and it is expected failures in the communication due to intermittence, interferences, intentional jamming, environment factors or partition of node groups.

The commanders should automatically know their neighbors and they may be able to send messages to Teams using the available neighbors (H1, B1 and B2). However, some of these nodes, or even all, may be unreachable or may not be adequate nodes to forward the data. Commander nodes should select the adequate intermediate nodes to forward the data to the respective destinations.

3.2.3 Population Support

The Population Support Scenario happens in catastrophe cases where it is important to offer support to population. Here there is one central node that acts as a coordination point (MS) for other nodes, e.g. ambulances or marines as shown in Figure 3.3. The topology is

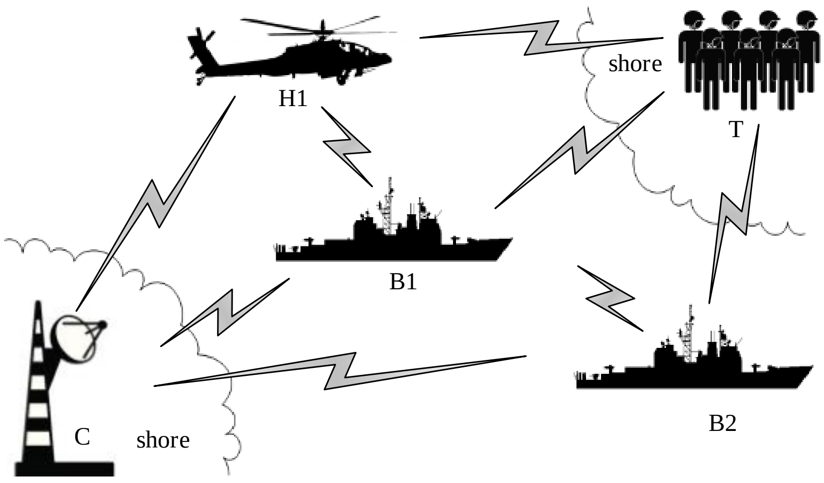


Figure 3.2: Naval and Amphibious Scenario

possibly static but some nodes may present mobility, and the teams may possibly connect to the Internet to communicate with the central coordinator, if necessary. However, the infrastructure may be destroyed or may be not always available due to scenario constrains.

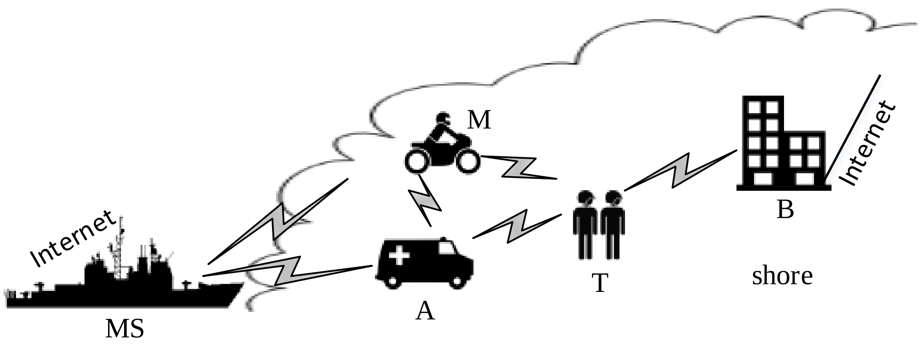


Figure 3.3: Population Support Scenario

It is necessary that rescue teams (T) maintain communications with the MS, either via other nodes (A or M), either via Internet access at B when possible.

3.2.4 Summary

The military scenarios presented in this section present generally the same issues, intermittence and interferences in the media and the possibility of some nodes to disconnect during long time intervals, due to mobility or connection failure. However, in all scenarios there are several spatial topology differences which distinguish them and the routing actions to be made.

It is important to guarantee communication between the nodes using whenever possible the links with better transmission quality, and provide routing protocols with stable behaviors.

3.3 Proposed Solution

This section presents a definition for the link quality, i.e. a quantitative function to define the quality of the wireless links. Then it presents the Q-PRoPHET, a quality-based routing protocol for wireless DTNs in disruptive environments, such as military or navy scenarios. This routing protocol uses the quality of the links to perform routing decisions under challenged scenarios, and it tries to avoid the poor links to send/forward data.

First it is presented the concept of quality which defines the link quality, and then it is presented the Q-PRoPHET routing protocol.

3.3.1 Link Quality

To define the link quality of a wireless link, it is necessary to establish a relation between what is the quality and how the quality may be measured. The link quality in this dissertation is defined as a dimensionless variable that aims to describe the capability of the link to exchange data with efficiency and effectiveness in wireless networks. However, to evaluate the quality and measure it, it is necessary to define and use numerical metrics with a qualitative relation with the links quality.

This work considers two numerical metrics to evaluate the quality of the links, the Signal Strength Intensity in dB and the connections' stability.

The SSI is the decibel (dB) difference between RF signal power measured at the receiving antenna and a fixed reference. *"Intensity like signal strength will generally fall off with distance from the source, although it also depends on the local conditions and the pathway from the source to the point."(Charles Richter, 1971).* Despite this quote is from seismology research area, it is applicable to characterize the SSI. It is expectable that SSI decreases with the distance due to signal propagation, although it may decrease due to other physical conditions that affect the RF propagation signal. Consequently this metric is good to represent the links quality, because it gives a value to the strength of the signal and consequently the proximity of the nodes, but it may not be sufficient because it evaluates "the strength of the received packets" and does not consider if they are all received, i.e., it is possible to have lost packets and the node will never know that the packets were lost.

The Link Stability (LStab) of the connections is capable to evaluate the success of received packets, i.e. it measures the ratio between successfully received packets (beacons) from a node and all emitted packets (beacons) from that node. This metric considers only the most recent packets (considering a certain temporal window) to define a stability associated to the moments closer to the decision time. The LStab measured at node A, between nodes A and B, for the most recent temporal window is defined according to equation 3.1.

$$LStab(A,B) = \frac{Nb_rA}{Nb_eB} \tag{3.1}$$

where Nb_rA is the number of received beacons in node A, and Nb_eB is the number of emmited beacons in node B. These two metrics allow to evaluate the quality of the connections. The SSI evaluates the "strength" of the connection and LStab evaluates the success of the communications, i.e. the quantity of received packets compared to the emitted packets in the closer moments to the decision time.

To define a complete link quality metric between two nodes, A and B, it is proposed the quality function present in equation 3.2, where the $f_{quality}(A,B)$ is the quality factor (link's quality), measured between nodes A and B; $LStab(A,B)$ is the stability normalized in the interval [0, 1]; $SSI(A,B)$ is the SSI in dB limited to SSI_{max}. The SSI_{max} is a constant and it represents the maximum expected SSI in good conditions of transmission, i.e. when the nodes are relatively closer to each other and with no interference.

$$f_{quality}(A,B) = \frac{LStab(A,B)}{2} + \frac{SSI(A,B)}{2 \times SSI_{max}} \tag{3.2}$$

In equation 3.2 the minimum value of $f_{quality}(A,B)$ is zero and the maximum value is one, due the fact that the two equation members, $\frac{Stab(A,B)}{2}$ and $\frac{SSI(A,B)}{2 \times SSI_{max}}$, are chosen in order to vary at most the quality factor by a 0.5 factor each. The representation of the quality as a function of the normalized stability and the normalized SSI is presented in the graph of Figure 3.4, where it is possible to confirm the minimum and maximum possible values of the quality factor.

3.3.2 Quality-PRoPHET

The Q-PRoPHET is a quality-based routing protocol originally based on PRoPHET protocol. Q-PRoPHET performs routing decisions based on the neighbors link's quality instead of the number of encounters, like PRoPHET (see section 2.3.2.2 or pages 12 to 15 of RFC6693 [39]). Besides, Q-PRoPHET is capable also to perform multi-hop routing decisions based on the intermediate links' qualities of the path. The main objective of this new routing protocol is to perform routing decisions to forward data using the best quality paths, instead of bad paths, i.e. paths with links characterized by weak signal or interferences whose may affect the success of the transmissions. These unwanted behaviors may happen in disruptive scenarios due to the fact that these environments are characterized by high quantity of short term connections. The usual DTN routing protocols may not be adequate to this type of environments due to the fact that the short-term connections may not be noticed by the protocols and may affect its performance.

Q-PRoPHET uses the quality factor, $f_{quality}$, function of SSI and LStab as defined in equation 3.2, to perform routing decisions based on these qualities.

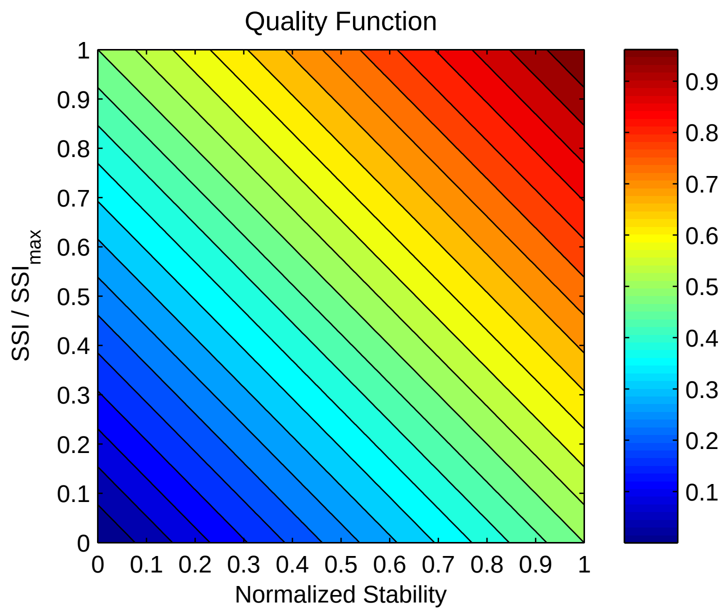


Figure 3.4: Quality Function

The operation mode of Q-PRoPHET is based and similar to the operation mode of PRoPHET protocol that routes the information based in a probability of a certain node to deliver information to the destination, either directly or indirectly, i.e. using multiple hops. This probability is increased at each encounter (equation 2.1) and decreases along the time (equation 2.2) without contacts. There is also a transitive probability (equation 2.3) to calculate a probability to deliver information to its destination through intermediate nodes (multi-hop). There may be applied different forwarding strategies, but the most common ones are GRTR and GTMX, previous stated in section 2.3.2.2. These equations associated with the forwarding strategy were replicated and modified in order to create a new quality-based routing protocol which uses the links' quality to perform routing decisions as follows.

Q-PRoPHET increases the quality associated to a certain destination based on equation 3.3, decreases the quality according to equation 3.4 and it may also calculate a quality associated to a path formed by specific nodes according to equation 3.5. The following paragraphs present a more detailed description related to these equations.

The Direct Contact (equation 3.3) updates the probability to deliver data with success based on the links quality. A good link has a high quality value, and a bad link has a low quality value. The probability to deliver data with success is the quality factor $f_{quality}$. Thus, Q-PRoPHET avoids to send replicas to poor quality links, i.e. surrounding nodes which the link's quality is lower compared to the local node.

$$P_{(A,B)} = f_{quality}(A, B) \tag{3.3}$$

The Decay Over Time (equation 3.4) is used to decrease the probability/quality along the time in the case that the node is unreachable. Variable γ is an aging constant that affects how fast the probability/quality decreases along the time, and k is the number of time units that have elapsed since the last time the metric was aged. This property is important in the case that it is needed to carry data between two other nodes that are not reachable at the same time and there are no other paths to route the information. For instance, if node A established contact recently with node B, there is the possibility that node A will establish contact with node B again. If a node C never established contact with node A, but established contact with node B, it should consider to deliver to B the bundles addressed to node A.

$$P_{(A,B)} = P_{(A,B)_{old}} \times \gamma^k \tag{3.4}$$

The Transitive Rule (equation 3.5) is based on the fact that, if a node A is in contact with node B and this node is in contact with node C, it may be a better choice to send data from node A to node C using node B if this node has better quality to deliver the message. The objective is to benefit a better quality transitive path than a low quality path. The graph of the quality associated to a path composed between two links (link 1 and link 2) is presented in Figure 3.5, where it is possible to verify that the ideal situation is when both links have unitary quality, and the worst situation is when at least one of the links has null quality which is sufficient to stop the communication. It is possible to verify according to the graph that the multiplication of the qualities of each edge allows to obtain a good representative of the quality of all transitive paths because, if all edges are good, the quality will be good, but if at least one edge is bad, the transitive quality will be bad.

$$P_{(A,C)} = Max(f_{quality}(A,C), f_{quality}(A,B) \times P_{(B,C)_{recv}} \times \beta) \tag{3.5}$$

However, unlike PRoPHET's GRTR and GTMX forwarding strategies, this routing protocol does not compare directly local node's quality to the destination with neighbors' quality to the destination, because it should consider also its own quality to the neighbor nodes to evaluate if it is a good decision to forward the data for a certain neighbor or not. Thus, it is necessary to perform changes also in the forwarding strategies. The two modified strategies are, considering that nodes A and B encounter each other, node D is the destination node and A contains a bundle destined to node D:

- **GRTR** just sends the bundle to node B if $P_{recv}(B,D) \times f_{quality}(A,B) \geqslant P_{(A,D)}$, i.e. if the probability/quality between node B and D times the local node's quality to node B is larger or equal than the local node's quality to node D, considering the quality of the path;
- **GTMX** just sends the bundle to node B if $P_{recv}(B,D) \times f_{quality}(A,B) \geqslant P_{(A,D)}$ and $NF > NF_{max}$, i.e. if the probability/quality between node B and D times local node's quality to node B is larger or equal than the local node's quality to node D, considering the quality of the path, and if the NF at node A is lower than the limit

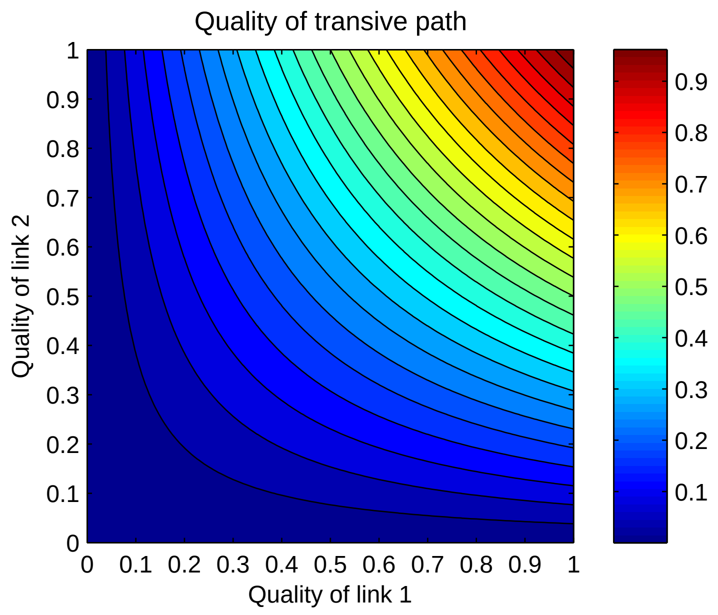


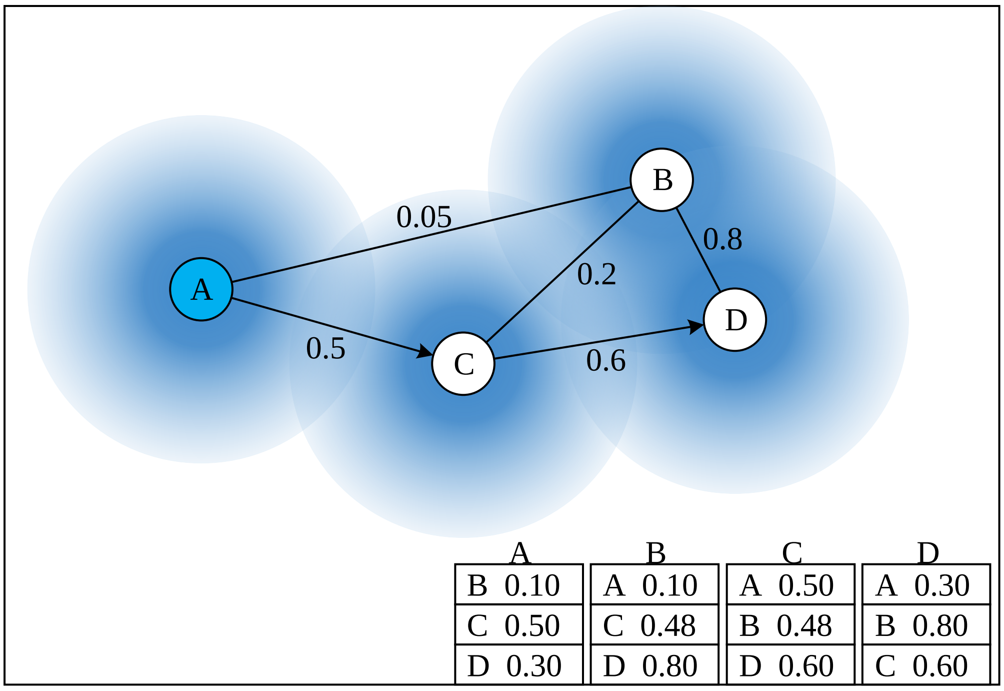
Figure 3.5: Quality of transitive path

NF_{max}. The NF is a method to control the number of replicas of a certain bundle in the network.

The proposed forwarding strategy allows to benefit the good transitive paths instead of a direct path, or even allows communication if the only available path is the transitive path.

To exemplify the Q-PRoPHET operation, Figure 3.6 illustrates a situation where node A has a bundle to deliver to node D, and there are no direct contacts between them. The quality tables contain the values of the best qualities of the links correspondent to each node established contact to other nodes, either the quality is associated to a direct path, or it is associated to a transitive path.

Node A has a bundle to deliver to node D which is not in direct contact. So, it will forward the bundle to neighbor nodes. Node A establishes contact with node D with an associated quality of 0.30, which is the best quality selected from all possible paths to the destination node. Node A then compares its quality with the quality of the paths via node B and via node C (separately). Via node B the quality has a value of 0.04, which is lower than the local quality, and then node A does not send the bundle to node B. Via node C the quality is 0.3, which is equal to the local quality, and according to the forwarding strategy (either GRTR or GTMX), the bundle will be sent to node C. After that, node C will verify that its direct path to node D has a quality of 0.60, which is larger than the quality via node B with the value of 0.16. Consequently, node C delivers directly the bundle to node D.



A	B	C	D
B 0.10	A 0.10	A 0.50	A 0.30
C 0.50	C 0.48	B 0.48	B 0.80
D 0.30	D 0.80	D 0.60	C 0.60

Figure 3.6: Q-PRoPHET Example

3.4 Chapter Considerations

This chapter presented the navy scenarios which present wireless communication challenges and the solution to improve the communication in those scenarios.

First, it was presented a definition of the quality to evaluate the wireless links based on the SSI and LStab. Then, it was presented the Q-PRoPHET routing protocol which uses the quality of the links, and a transitive property to perform routing allowing multi-hop forwarding.

The next chapter will describe the implementation of an API to measure the quality of the links, and the implementation of the Q-PRoPHET in IBR-DTN, which uses this API to get the quality values.

Chapter 4

Architecture and Implementation

4.1 Introduction

This chapter presents the Quality Connection Reader (QCR) API and the Q-PRoPHET architecture and implementation.

Section 4.2 explains the process that QCR uses to gather the quality parameters (SSI and *LStab*) and how to maintain these parameters to calculate the quality of the wireless links.

Section 4.3 presents the modifications performed in IBR-DTN to insert the Q-PRoPHET routing protocol and to communicate with QCR.

Section 4.4 presents the chapter summary and considerations.

4.2 Quality Connection Reader

The QCR is an API that gathers the quality parameters and calculates the links' quality for DTN neighbors, as well as to maintain an updated list of the parameters associated to each neighbor. This module offers a communication interface to easily integrate it with a DTN platform or other applications that need to measure the quality of the links without engaging in implementation details, e.g. AP selection based on the quality of the links.

4.2.1 Access the Quality Parameters

The quality parameters are the SSI and the links' stability, as stated in section 3.3.1.

To gather the SSI, it is used the SSI value presented in Radiotap Header (see Figure 2.25 from section 2.4.4) obtained from the network card's driver. This process involves to capture frames to access radiotap's SSI values and frame's source address, i.e. the MAC address related to identification purposes. To access these fields it is used a packets capture (*libpcap*) in a virtual monitor interface created for this purpose, as mentioned in section 2.4.5.2.

To calculate the stability value it is necessary a method to each node to know that it is missing packets. To estimate that, it is not necessary and practicable to consider all packets. Thus, it may be used a specific type of packets/frames as an approach to estimate the stability value, which are the advertising beacons where each node sends periodic beacons containing a SN. The SNs are incremented in each new beacon and the other nodes are able to listen these beacons if they are reachable. Consequently, the other nodes may know how many beacons were received and how many beacons were lost by analyzing the beacons' SN. Thus, a node can calculate the stability value as the ratio between the listened beacons and all emitted beacons from a specific neighbor, in a recent temporal window in order to consider a stability of the actual moment.

Beacons may be implemented, but it is possible to use the IEEE 802.11 Beacon frames, which are beacons periodically sent by the nodes' MAC layer, as stated in section 2.4.3, to announce its presence to neighboring nodes. Beacons contains information about the source node, including its MAC address and a SN. Thus, they meet the requirements to be the option to evaluate the LStab. To read these beacons can be used libpcap to perform packet capture. However, in this work the beacons are implemented by the QCR due to reason that will be explained later.

4.2.2 Architecture

To gather the quality factors from DTN neighbors and to integrate these qualities with a DTN platform or other program, it was developed the Quality Connection Reader (QCR) module in *C++* programming language. QCR is capable of tracking and filtering the beacons from DTN neighbors and extract the relevant information (in this case beacons' SN and radiotap's SSI) to calculate the quality factor presented in section 3.3.1.

The QCR architecture and interactions with the DTN platform are presented in Figure 4.1.

Unlike the proposed IEEE 802.11 beacon frames in section 3.3.1, the presented QCR module implements its own beacons system, due to the fact that the pcap performance to capture IEEE 802.11 beacon frames in monitor mode varies significantly according to the platform being used [92]. It was implemented a first version of QCR using IEEE 802.11 beacon frames, but it presented problems on the experimental testbed because not all beacons were being captured, even when the nodes were closer to each other. The pcap captures (depending on the board's performance) all packets in the media, including all Wireless Fidelity (Wi-Fi) nodes, not only the DTN nodes, because the interface captures in monitor mode.

QCR is able to communicate with a DTN platform via an Inter Process Communication (IPC) socket. QCR is easy to integrate with different DTN implementations, like IBR-DTN (section 2.2.6.2) or DTN2 (section 2.2.6.1), or other applications that need to measure the quality of the links, e.g. access point selection based on the link quality.

There are many commands that DTN platform can send to the QCR, but the most important ones are `Add new EID` to track quality; `Remove EID` and `Request quality by EID`. Here, the DTN platform acts like a master program that sends instructions and

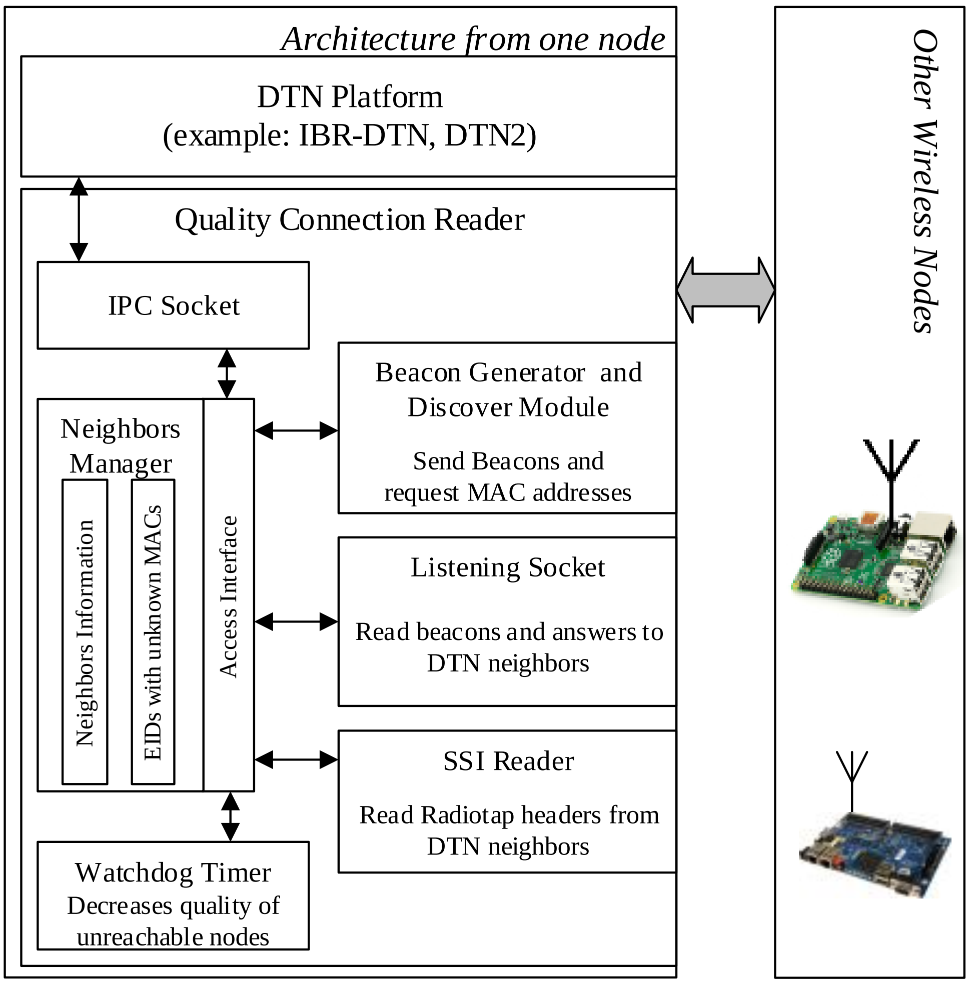

Architecture from one node
DTN Platform
(example: IBR-DTN, DTN2)
Quality Connection Reader
IPC Socket
Beacon Generator and Discover Module
Send Beacons and request MAC addresses
Neighbors Manager
Neighbors Information
EIDs with unknown MACs
Access Interface
Listening Socket
Read beacons and answers to DTN neighbors
SSI Reader
Read Radiotap headers from DTN neighbors
Watchdog Timer
Decreases quality of unreachable nodes
Other Wireless Nodes

Figure 4.1: QCR: Block Diagram and Interactions

receives information to/from the QCR, the slave program. To perform its tasks, QCR interacts with different layers from the network stack as presented in Figure 4.2. QCR gathers information from PHY and MAC layer (in Data Link) to obtain the metrics to calculate the links quality and to identify the associated neighbors by MAC address. It uses also the transport layer to communicate with other nodes' QCR and implements its own beacons system; it also communicates with IBR-DTN (Figure 4.1), which is the master program in this case, but it can be another application.

However, to integrate the QCR in IBR-DTN, the chosen DTN platform, it is necessary to perform changes in the IBR-DTN source code, relatively to all code dimension, in order to add a new routing protocol and prepare it to communicate with the QCR via an IPC socket using a set of rules defined in both sides.

The following topics present the QCR implementation and operation details.

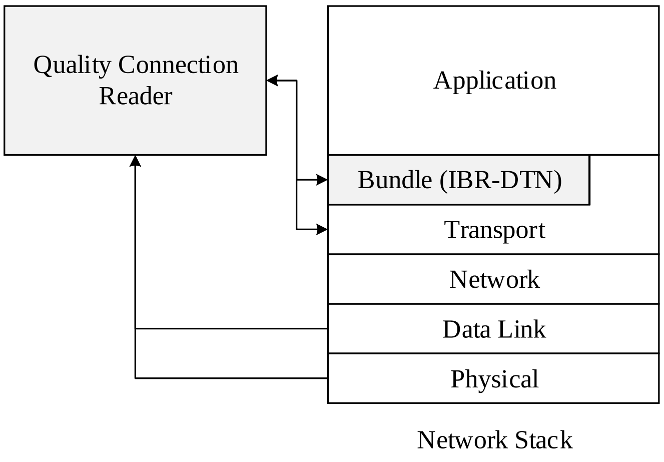


Figure 4.2: Relation between QCR and network stack

4.2.2.1 Initialization

The QCR initialization is presented in Figure 4.3.

QCR begins by creating a *Neighbors Manager* which is a shared structure with the purpose to store all relevant information about DTN neighbors. More, details about this structure are presented in section 4.2.2.2.

Then, the QCR launches four threads that perform the core tasks: the Periodic Events, the Listening Socket, the SSI Reader and the IPC Socket. These threads may access concurrently the Neighbors Manager Structure to read or to change data. Usually, all threads are in an infinite loop cycle to perform their tasks. Consequently they will not finish, because this is not the objective.

4.2.2.2 Neighbors Manager

Neighbors Manager [`neighborsManagement`] (Figure 4.4) is a shared structure with a public access interface to handle concurrency between different threads. This structure includes several objects and methods to maintain a list of Neighbors Information [`neighborList`] and a list of EIDs which MACs are unknown [`unknownNeighbors`]. It also contains a string with its own EID called `myEID`. Two vectors containing node's monitor interfaces (`myMonitors`) and MAC addresses (`myMACs`) from its own interfaces used by DTN platform are also present in the Neighbors Manager, and their purpose will be explained later in the end of this section. Besides that, it contains a set of `mutexs` to apply mutual access exclusion to each one of the stated structures or objects by different threads operating concurrently.

Figures 4.4 and 4.5 present the class diagram of Neighbors Manager, but due to the lack of space in the page, the diagram needs to be presented in two pages.

There is an extensive list of methods. An overview about classes will be presented and the most important attributes and methods will have a more detailed description.

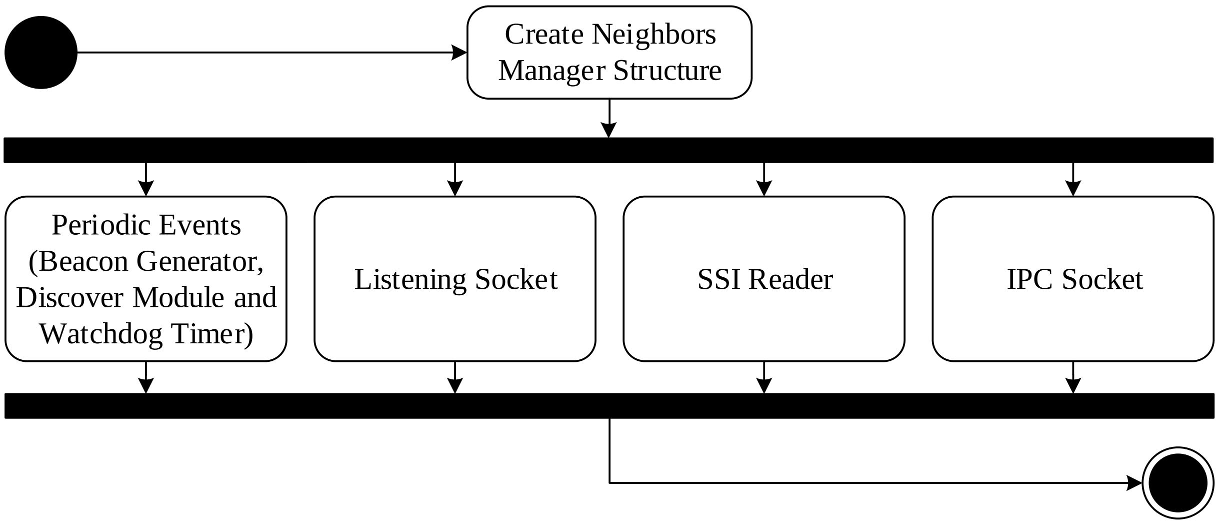

Create Neighbors Manager Structure
Periodic Events (Beacon Generator, Discover Module and Watchdog Timer)
Listening Socket
SSI Reader
IPC Socket

Figure 4.3: QCR: Initialization Flow Chart

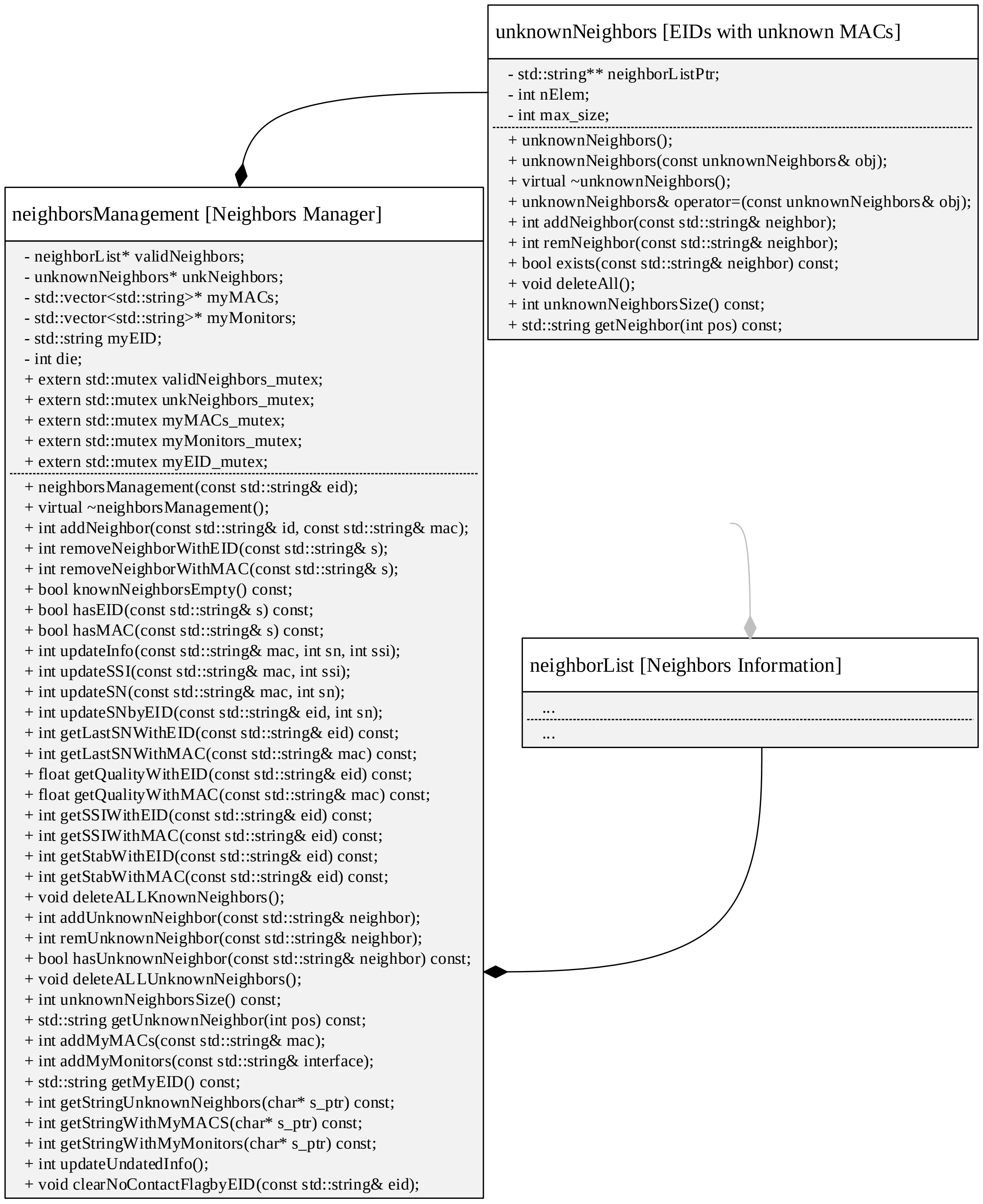
`basicParam` (basic parameter) is an abstract base class[1] and just contains one attribute, the quality value. It contains a pure virtual method[2] to calculate and update the quality (`calcQuality()`) and it contains a method to read quality value (`getQuality()`). Besides that, `basicParam` contains another pure virtual method to identify the associated interface type, for example the Wi-Fi, the only interface implemented so far. `basicParam()` and ∼`basicParam()` are the constructor and the destructor of the class, respectively.

`wifi80211params` (Wi-Fi 802.11 parameters) is a derived class from `basicParam` base class. `wifi80211params` contains information about the parameters associated to a Wi-Fi interface. These parameters allow to calculate the quality value for that type of interface (for Wi-Fi these parameters are the LStab and the SSI).

The stability value is not directly represented in `wifi80211params` class due to the fact that it depends on the last received beacons. Thus, there are some objects to maintain the necessary data related with the received beacons. `recentSN_fifo` is a pointer to a bitmap, which together with `recentSN_ptr` and `recentSN_value`, represent which beacons were received in the recent temporal window. The number of recent beacons under evaluation is defined by `SN_fifo_size`, which is measured in sets of four bytes (32 bits). Each bit in `recentSN_fifo` means that a certain beacon was received or not. Figure 4.6 presents an

[1]Abstract base classes are classes that can be only used as base classes, i.e., it is not possible to instantiate objects from them.

[2]A pure virtual method is a method defined in the abstract base class, but it is only implemented in a derived class. Besides, a class with at least one pure virtual method is an abstract class.

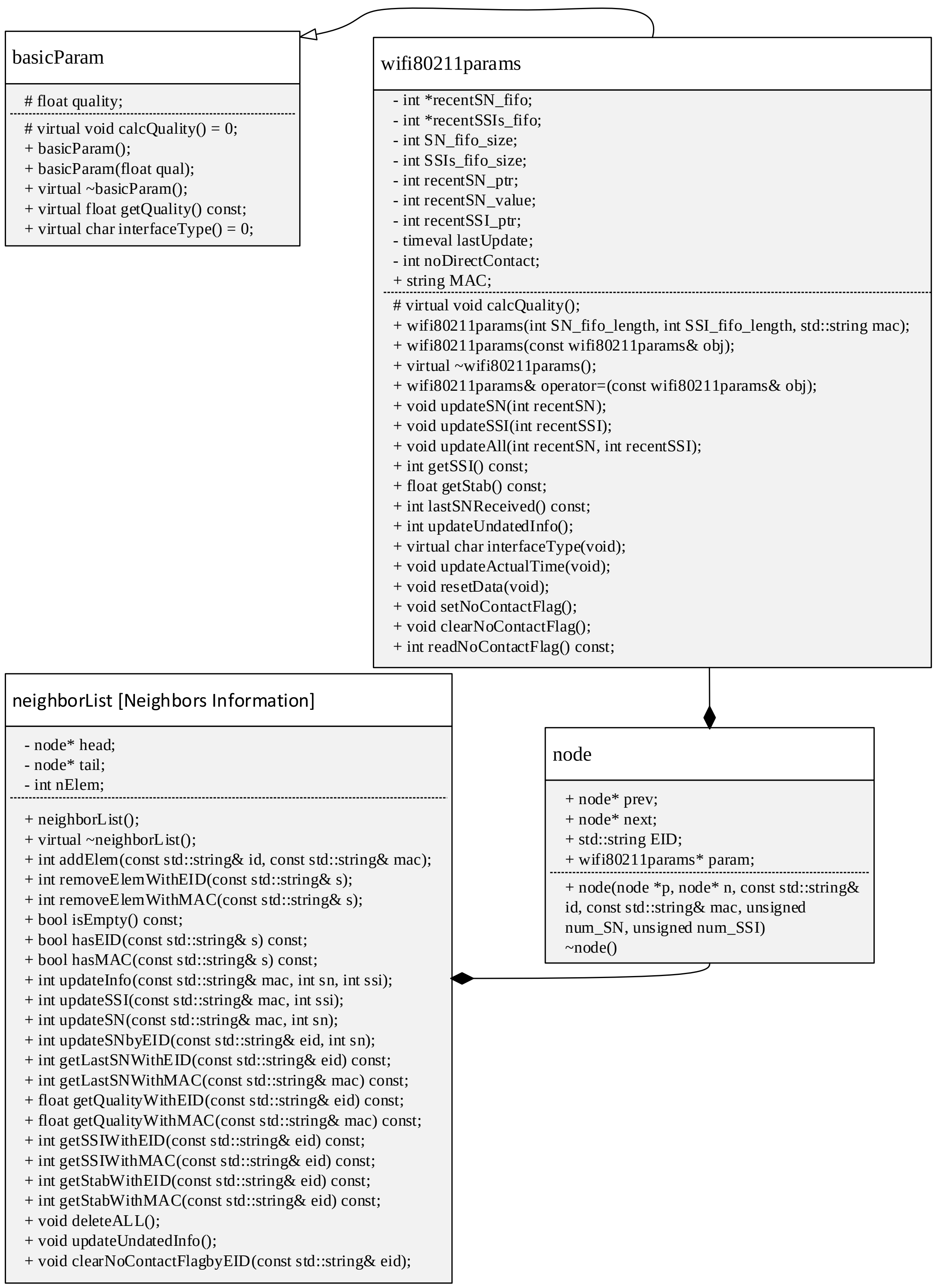


```
unknownNeighbors [EIDs with unknown MACs]
- std::string** neighborListPtr;
- int nElem;
- int max_size;
---------------------------------------------
+ unknownNeighbors();
+ unknownNeighbors(const unknownNeighbors& obj);
+ virtual ~unknownNeighbors();
+ unknownNeighbors& operator=(const unknownNeighbors& obj);
+ int addNeighbor(const std::string& neighbor);
+ int remNeighbor(const std::string& neighbor);
+ bool exists(const std::string& neighbor) const;
+ void deleteAll();
+ int unknownNeighborsSize() const;
+ std::string getNeighbor(int pos) const;
```

```
neighborsManagement [Neighbors Manager]
- neighborList* validNeighbors;
- unknownNeighbors* unkNeighbors;
- std::vector<std::string>* myMACs;
- std::vector<std::string>* myMonitors;
- std::string myEID;
- int die;
+ extern std::mutex validNeighbors_mutex;
+ extern std::mutex unkNeighbors_mutex;
+ extern std::mutex myMACs_mutex;
+ extern std::mutex myMonitors_mutex;
+ extern std::mutex myEID_mutex;
---------------------------------------------
+ neighborsManagement(const std::string& eid);
+ virtual ~neighborsManagement();
+ int addNeighbor(const std::string& id, const std::string& mac);
+ int removeNeighborWithEID(const std::string& s);
+ int removeNeighborWithMAC(const std::string& s);
+ bool knownNeighborsEmpty() const;
+ bool hasEID(const std::string& s) const;
+ bool hasMAC(const std::string& s) const;
+ int updateInfo(const std::string& mac, int sn, int ssi);
+ int updateSSI(const std::string& mac, int ssi);
+ int updateSN(const std::string& mac, int sn);
+ int updateSNbyEID(const std::string& eid, int sn);
+ int getLastSNWithEID(const std::string& eid) const;
+ int getLastSNWithMAC(const std::string& mac) const;
+ float getQualityWithEID(const std::string& eid) const;
+ float getQualityWithMAC(const std::string& mac) const;
+ int getSSIWithEID(const std::string& eid) const;
+ int getSSIWithMAC(const std::string& eid) const;
+ int getStabWithEID(const std::string& eid) const;
+ int getStabWithMAC(const std::string& eid) const;
+ void deleteALLKnownNeighbors();
+ int addUnknownNeighbor(const std::string& neighbor);
+ int remUnknownNeighbor(const std::string& neighbor);
+ bool hasUnknownNeighbor(const std::string& neighbor) const;
+ void deleteALLUnknownNeighbors();
+ int unknownNeighborsSize() const;
+ std::string getUnknownNeighbor(int pos) const;
+ int addMyMACs(const std::string& mac);
+ int addMyMonitors(const std::string& interface);
+ std::string getMyEID() const;
+ int getStringUnknownNeighbors(char* s_ptr) const;
+ int getStringWithMyMACS(char* s_ptr) const;
+ int getStringWithMyMonitors(char* s_ptr) const;
+ int updateUndatedInfo();
+ void clearNoContactFlagbyEID(const std::string& eid);
```

```
neighborList [Neighbors Information]
...
---------------------------------------------
...
```

Figure 4.4: QCR: Neighbors Manager [`neighborsManagement`] Class Diagram


```
basicParam
# float quality;
# virtual void calcQuality() = 0;
+ basicParam();
+ basicParam(float qual);
+ virtual ~basicParam();
+ virtual float getQuality() const;
+ virtual char interfaceType() = 0;
```

```
wifi80211params
- int *recentSN_fifo;
- int *recentSSIs_fifo;
- int SN_fifo_size;
- int SSIs_fifo_size;
- int recentSN_ptr;
- int recentSN_value;
- int recentSSI_ptr;
- timeval lastUpdate;
- int noDirectContact;
+ string MAC;
# virtual void calcQuality();
+ wifi80211params(int SN_fifo_length, int SSI_fifo_length, std::string mac);
+ wifi80211params(const wifi80211params& obj);
+ virtual ~wifi80211params();
+ wifi80211params& operator=(const wifi80211params& obj);
+ void updateSN(int recentSN);
+ void updateSSI(int recentSSI);
+ void updateAll(int recentSN, int recentSSI);
+ int getSSI() const;
+ float getStab() const;
+ int lastSNReceived() const;
+ int updateUndatedInfo();
+ virtual char interfaceType(void);
+ void updateActualTime(void);
+ void resetData(void);
+ void setNoContactFlag();
+ void clearNoContactFlag();
+ int readNoContactFlag() const;
```

```
neighborList [Neighbors Information]
- node* head;
- node* tail;
- int nElem;
+ neighborList();
+ virtual ~neighborList();
+ int addElem(const std::string& id, const std::string& mac);
+ int removeElemWithEID(const std::string& s);
+ int removeElemWithMAC(const std::string& s);
+ bool isEmpty() const;
+ bool hasEID(const std::string& s) const;
+ bool hasMAC(const std::string& s) const;
+ int updateInfo(const std::string& mac, int sn, int ssi);
+ int updateSSI(const std::string& mac, int ssi);
+ int updateSN(const std::string& mac, int sn);
+ int updateSNbyEID(const std::string& eid, int sn);
+ int getLastSNWithEID(const std::string& eid) const;
+ int getLastSNWithMAC(const std::string& mac) const;
+ float getQualityWithEID(const std::string& eid) const;
+ float getQualityWithMAC(const std::string& mac) const;
+ int getSSIWithEID(const std::string& eid) const;
+ int getSSIWithMAC(const std::string& eid) const;
+ int getStabWithEID(const std::string& eid) const;
+ int getStabWithMAC(const std::string& eid) const;
+ void deleteALL();
+ void updateUndatedInfo();
+ void clearNoContactFlagbyEID(const std::string& eid);
```

```
node
+ node* prev;
+ node* next;
+ std::string EID;
+ wifi80211params* param;
+ node(node *p, node* n, const std::string&
id, const std::string& mac, unsigned
num_SN, unsigned num_SSI)
~node()
```

Figure 4.5: QCR: Neighbors Information [`neighborList`] Class Diagram

example to better understand the concept.

In Figure 4.6, the `SN_fifo_size` is 1. This means that stability is considering the last 32 beacons, which `recentSN_fifo` has only four bytes to maintain the reception information associated to the last 32 beacons. In Figure 4.6(a) the last received beacon has SN=1000 and it sets the bit 16 from the bitmap. The `recentSN_ptr=17` because it is the next position to update when a new beacon is received. The stability is 25/32, because 25 bits are set, i.e. in the last emitted 32 beacons, this node received 25 beacons. To count the set bits it is used an efficient algorithm to count the ones in a word [94], because this structure is regularly changing and the method to calculate the stability should be efficient to save time and CPU consumption. The used algorithm is presented in Algorithm 4.1 and it was adapted from C code example present in [95].

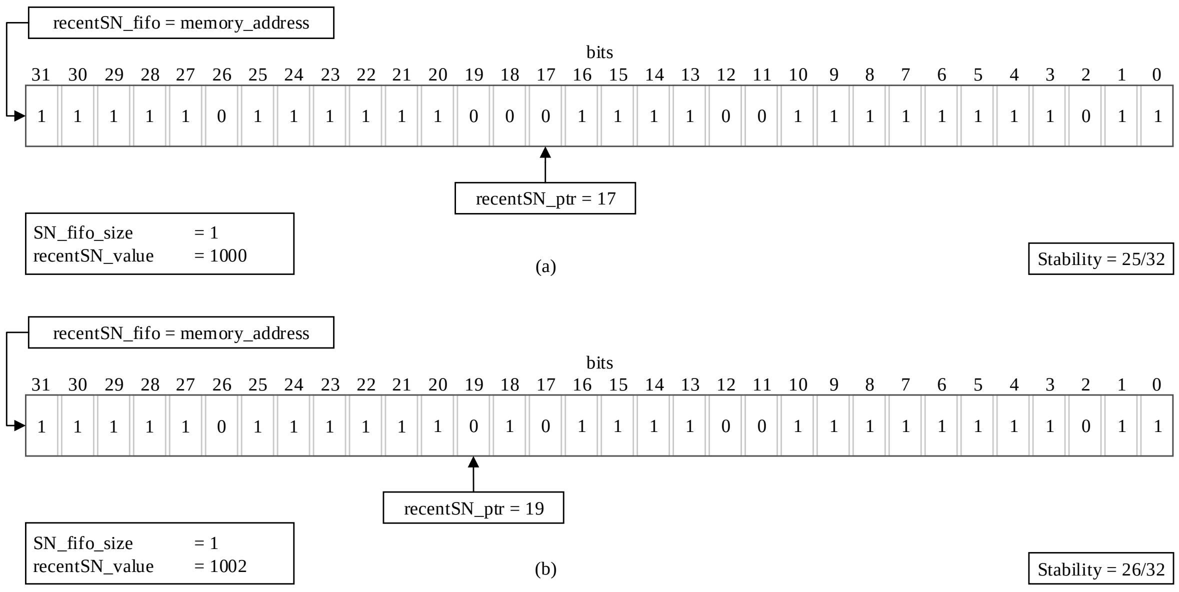


Figure 4.6: QCR: Sequence Numbers bitmap example

In Figure 4.6(b) it is received a new beacon with SN=1002. This means that the beacon with SN=1001 was missed. Thus, bit 17 is reset to consider beacon miss 1001 and bit 18 is set. After that, `recentSN_ptr`=19, because it will be the next expected beacon to be filled when a new beacon arrives.

In this case the value of `recentSN_ptr` varies between 0 and 31 due to the fact that there are only 32 bits.

The informations related to the SSI are maintained in `recentSSIs_fifo`, that is a pointer to an integer array that contains the most recent SSIs from received packets. The number of SSIs stored is defined by `SSIs_fifo_size`, and `recentSSI_ptr` is a pointer to the position where the next SSI will be stored. In Figure 4.13 the last SSI measured was 36dB. When a new value of SSI arrives, it will be stored in the position pointed by `recenSSI_ptr`, which is position 1. The value of considered SSI is the mean of all values

Algorithm 4.1 Calculate the number of one bits in a word

```
Input: $u$
Output: $c$
  $U \leftarrow u$
  $N \leftarrow (U >> 1)\&033333333333$
  $U \leftarrow (U - N)$
  $N \leftarrow (N >> 1)\&033333333333$
  $U \leftarrow (U - N)$
  $U \leftarrow (U + (U >> 3))\&030707070707$
  $c \leftarrow U\%63$
```

stored in the array pointed by `recentSSIs_fifo`. This methodology is adopted because it is possible that single isolated cases, such as a constructive reflection in the media, increases the SSI of one received packet, affecting strongly the quality value. Consequently, a routing decision could be affected, to avoid that, the SSI used to calculate the quality is the mean of the last three measures.

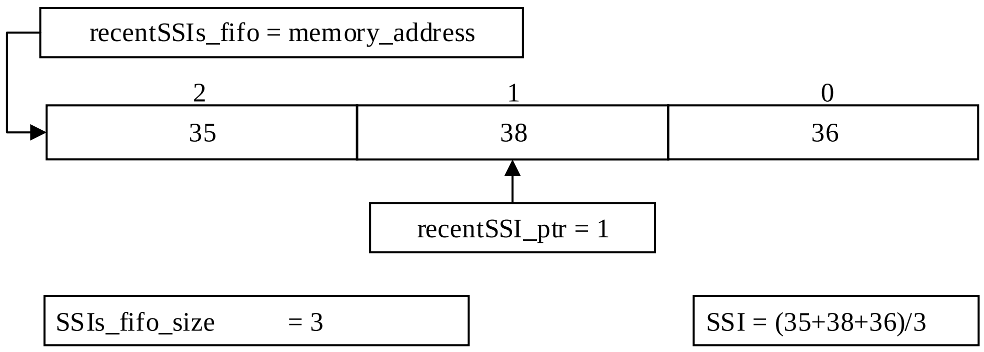


Figure 4.7: QCR: SSIs storage example

The `lastUpdate` variable present in `wifi80211params` stores the time of the last update of stability and SSI metrics. This time is useful to identify when a neighbor is out of reach and their parameters were not updated recently, and consequently, they should be decreased.

`MAC` attribute contains the neighbors interface's MAC address, i.e. the interface that the local node is tracking the quality by its MAC address.

`neighborList` class [Neighbors Information] contains a double-linked list of nodes. Each node of the list contains an EID identifying the neighbor and a pointer to `wifi80211params` object that contains the quality value and associated metrics. A visual representation of `neighborList` is presented in Figure 4.8, where `head` and `tail` are pointers to the double-linked list, and `nElem` is the number of nodes presented in the list. This list may contain information about different neighbors, each one identified by its EID. Each EID node contains a pointer to a `wifi80211param` object that contains parameters information, including the quality, and contains also the neighbor's MAC address.

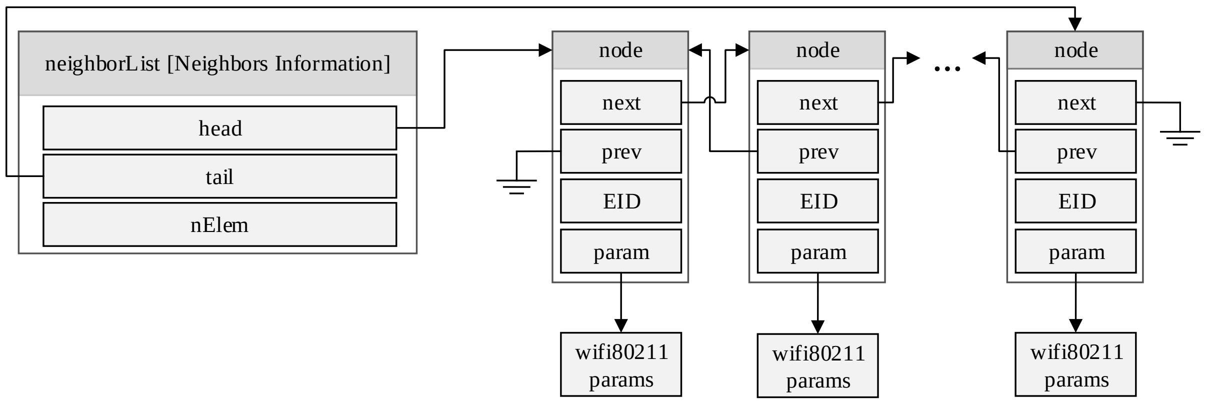


Figure 4.8: QCR: `neighborList` [Neighbors Information]

`unknownNeighbors` class [EIDs with unknown MACs] contains the known EIDs which MAC addresses are still unknown. This list is used mainly for the Discover Module to discover the MAC addresses from these EIDs. An example of the structure is presented in Figure 4.9 where `neighborListPtr` is a pointer to an array of pointers to strings containing the EIDs; `nElem` is the number of EIDs present in the list, and `max_size` is the length of the `unknownNeighbors` array that may be different of `nElem`, if the array is not full. If the array fills, the `max_size` can increase adaptively to a higher value, allowing the insertion of new unknown neighbors which EIDs are known but MACs are not.

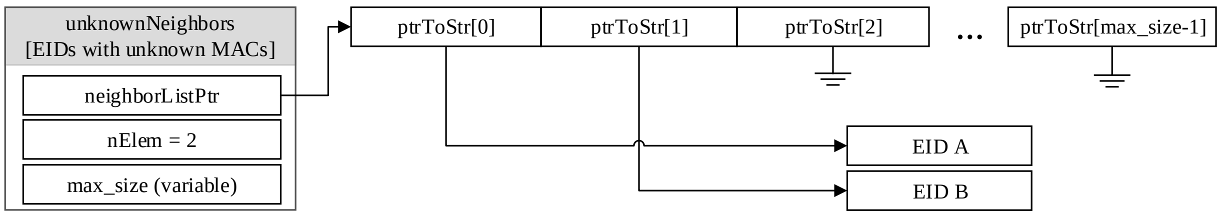


Figure 4.9: QCR: `unknwonNeighbors` [EIDs with unknown MACs]

`neighborsManagement` class [Neighbors Manager] contains the Neighbors Information List and the EIDs with unknown MACs List. It also contains a string with its own EID, a string vector with its own MAC interfaces, a string vector with its own network monitor interfaces and a variable (die) to signal the threads when the program should finish. These objects are accessed by different threads using a mutual exclusion mechanism. All public methods from `neighborsManagement` use the `mutexs` to perform their tasks.

4.2.2.3 QCR General Packet

The packet structure used by QCR to communicate with other QCR nodes is presented in Figure 4.10.

A `QCR App Identifier` is proposed whose function is to identify the application. The QCR applications use always the same UDP port, but if there are other nodes using the same port and they send broadcast messages, those messages will be received by QCR module. To prevent this situation, it is used identifiers to identify the application, not to authenticate it. However, in Wi-Fi this situation is not a problem because QCR operates inside a defined WLAN, where the nodes in the network are authenticated.

The `COMMAND` field indicates the purpose of the packet, i.e., the type of packet. The actual commands available are Beacon, Request MACs and Acknowledge MACs. Each type of command has a defined type of `Content`.

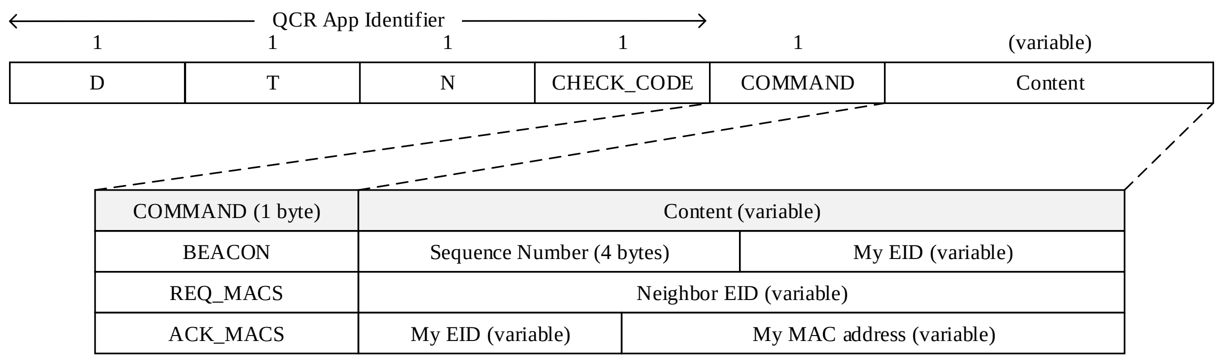


Figure 4.10: QCR General Packet

The content field of the `BEACON` command includes a SN and the source's EID of the beacon. The content of the Request MACs (`REQ_MACS`) command includes the neighbor's EID that are used to discover its MAC address. In the Acknowledge MACs (`ACK_MACS`), the content field includes the packet's source EID and the list of MAC addresses.

All packets are always sent with broadcast addresses. In the beacon packets the objective is to send them to all reachable neighbors. The objective of the Request MACs command is also to send the packet to all reachable neighbors because the sender node does not know the destination and it is trying to discover it. The main objective of the Acknowledge MACs command is to send the message (answer) to the query node, but to consider the situation where there may be more neighbors that need the same information, the packet is sent to the broadcast address, which may decrease the number of Request MACs in the network, e.g. in the early state of network formation where all nodes will discover the new neighbors and query them by its MAC wireless interface address.

4.2.2.4 Periodic Events: Beacon Generator, Discover Module and Watchdog Timer

The Periodic Events thread is responsible to run all periodical tasks: Beacon Generator, Discover Module and Watchdog Timer.

Beacon Generator is the module responsible to generate and send periodically broadcast beacons to the media with an associated SN between 0 and `BEACON_LIMIT`-1. The value selected for `BEACON_LIMIT` is 4096 based on the SN limit used by IEEE 802.11 beacon frames. The defined periodicity for the Beacon Generator is `PERIOD_MS`=200 (ms).

The Discover Module is responsible for periodically discover the MAC addresses from the neighbors that their EIDs are known, but their MACs are unknown. This module reads periodically the `unknownNeighbors` List and it sends periodic messages containing queries for EIDs which associated MACs are unknown. The defined periodicity for Discover Module is 1 second due the fact that this trade-off for the user is insignificant, but for the boards it is large enough to not overload them with this task.

Watchdog Timer is responsible to decrease the quality value from unreachable/non-contactable neighbors. When a neighbor is not in contact, there are no received beacons and SSI values. This means that it is necessary to decrease the quality because the node is not reachable. The adopted method consists in using a periodic Watchdog Timer[3] that, every 200ms, verifies the last time that the quality values were updated. If a quality value was not updated since `MS_WINDOW`=2 (seconds) the quality is updated to zero. The Flow Chart presented in Figure 4.11 exemplifies the Periodic Events flow.

In the beginning the `iterationCounter` and `beaconSN` variables are initialized with the value 0. Then, it is created an UDP socket running on port 12347. Then, the thread verifies if it should finish or continue to run. If the thread continues running, it creates a beacon packet with an associated `beaconSN` and sends it to the broadcast address. Then, the `beaconSN` value is increased by 1 and, if it reaches the `BEACON_LIMIT`, the value is reset to 0. Then, the Watchdog Timer runs and after that the `iterationCounter` is incremented. Thus, if the `iterationCounter` is equal to `TICKET_1SEC`, the Discover Module runs and it sends broadcast discovery packets to the network requesting the unknown MAC addresses from the EIDs list with unknown MACs. At this point the work for this iteration is finished and the thread sleeps during `PERIOD_MS`. Then, it verifies the finish condition to decide if it should continue running or if it should finish. The finish condition is verified by reading the `die` flag present in the `neighborsManagement` class. If the flag equals to '1', then the thread exits.

4.2.2.5 Listening Socket

The Listening Socket module has the function to listen the media and receive the QCR general packets (Figure 4.10) from another nodes' QCRs. According to the received packet, it may be necessary to update the local information, like neighbor's actual SN or add a

[3]A Watchdog Timer is a counter that is programmed with a specific time and it is always counting the time and when reaches the specified value, it resets and invokes a certain action

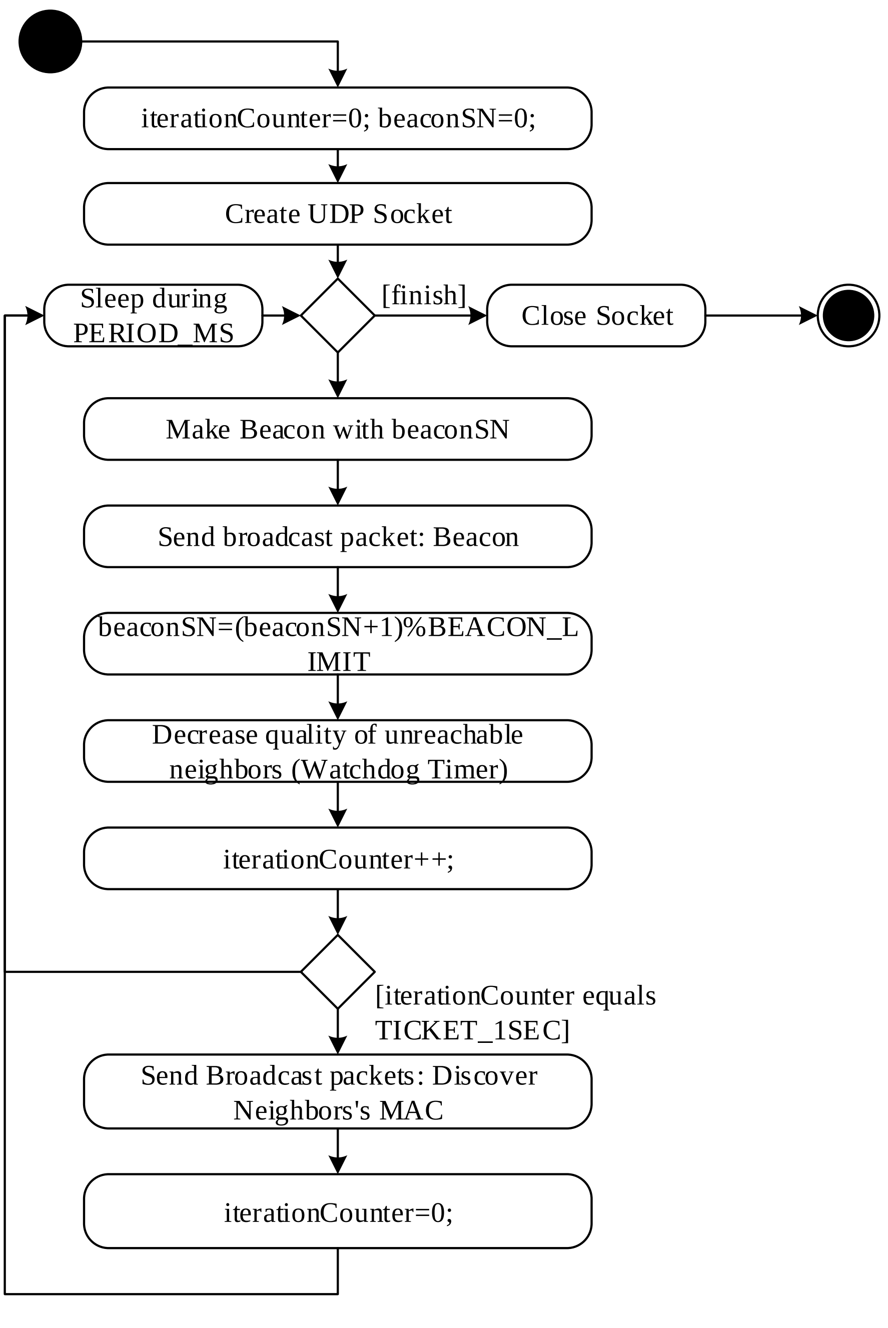

iterationCounter=0; beaconSN=0;
Create UDP Socket
Sleep during PERIOD_MS
[finish]
Close Socket
Make Beacon with beaconSN
Send broadcast packet: Beacon
beaconSN=(beaconSN+1)%BEACON_LIMIT
Decrease quality of unreachable neighbors (Watchdog Timer)
iterationCounter++;
[iterationCounter equals TICKET_1SEC]
Send Broadcast packets: Discover Neighbors's MAC
iterationCounter=0;

Figure 4.11: QCR: Periodic Events (Beacon Generator, Discover Module and Watchdog Timer) Flow Chart

---

pair EID/MAC or even to send a `ACK_MACS` packet to inform the neighbor about the local node's MAC addresses.

The Listening Socket flow presented in Figure 4.12 is the following: first it is created an UDP socket on port '12347', the same used by the Beacon Generator module. Then,

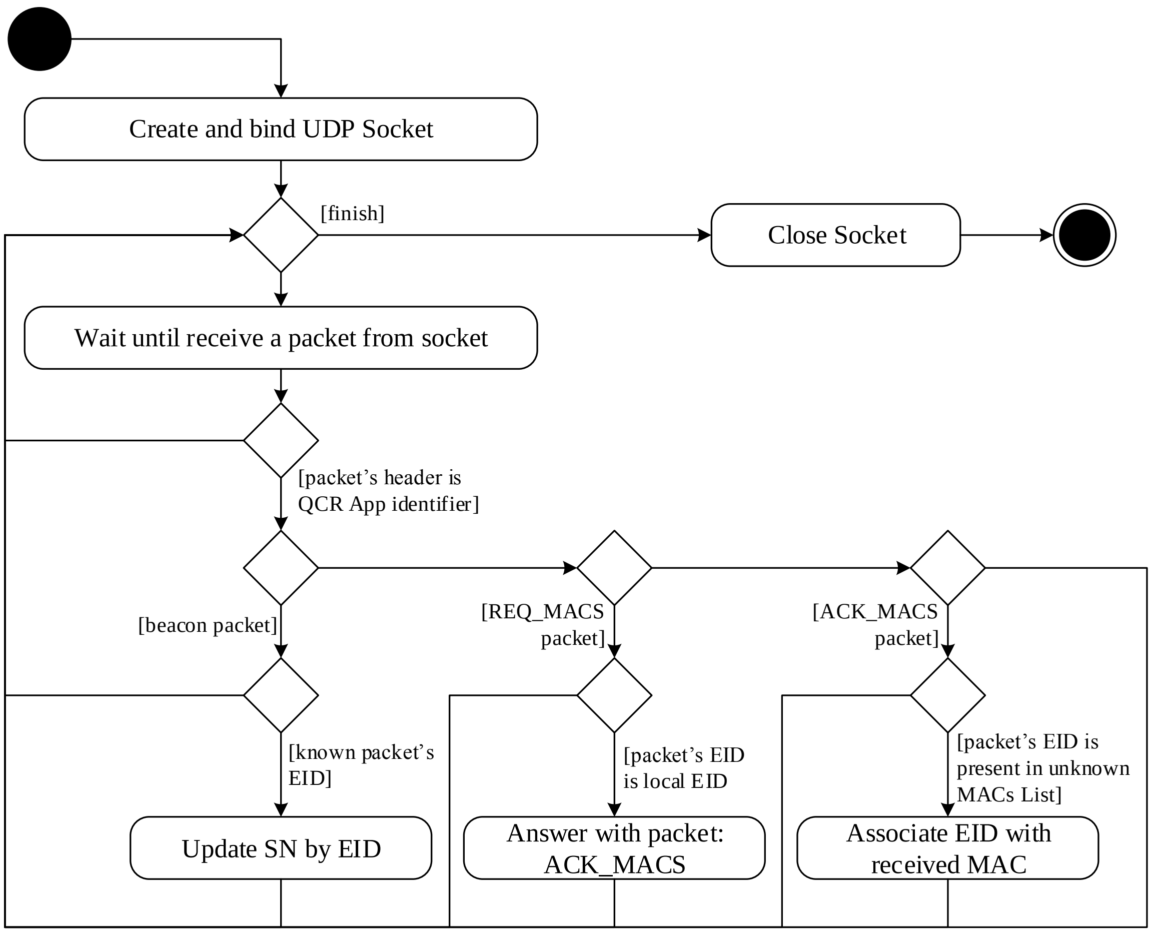
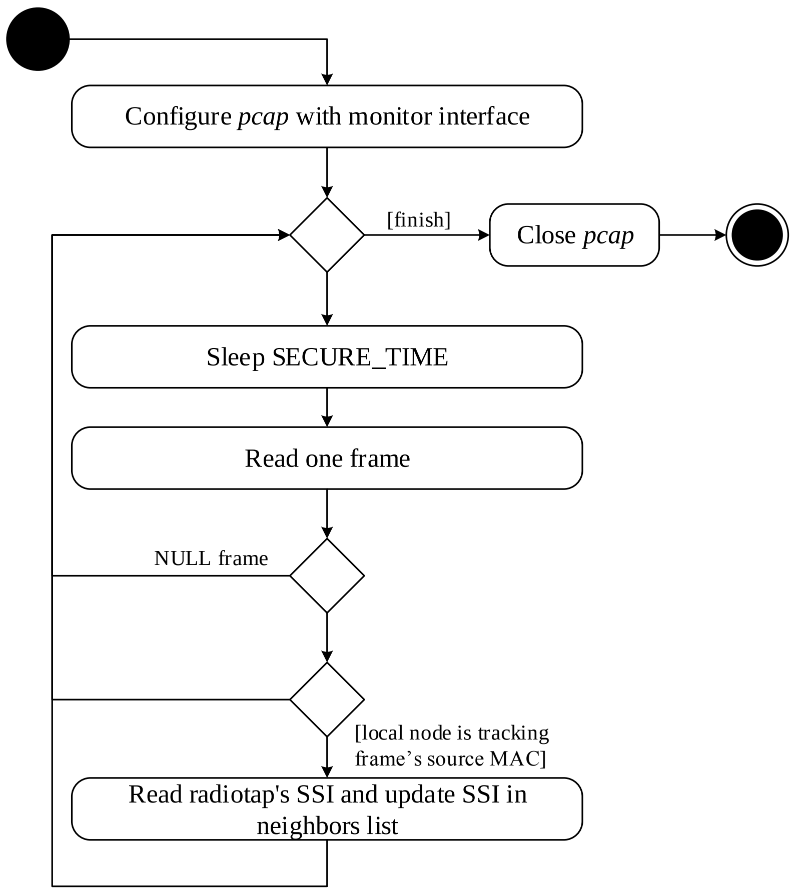

Create and bind UDP Socket
[finish]
Close Socket
Wait until receive a packet from socket
[packet's header is QCR App identifier]
[beacon packet]
[REQ_MACS packet]
[ACK_MACS packet]
[known packet's EID]
[packet's EID is local EID]
[packet's EID is present in unknown MACs List]
Update SN by EID
Answer with packet: ACK_MACS
Associate EID with received MAC

Figure 4.12: QCR: Listening Socket Flow Chart

the finish condition is tested and the thread continues its execution due to the fact that it is the first cycle. After that, the thread waits until it receives a new packet. When a new packet is received, the QCR App identifier is compared with the expected identifier. If they do not match, i.e. if they are different, the thread returns to finish condition and consequently it waits for a new packet. If they match, the thread reads the packet type. If the packet type is Beacon and if the packet's source EID is present in Neighbors List, the SN associated to this neighbor is updated. If the packet is a Request MACs and the EID present in the content field is the local node's EID, it sends a Acknowledge MACs with its MAC. However, if the received packet is an Acknowledgment MACs and if the packet's source EID is presented on EIDs with unknown MACs List, that EID and content MACs will be added to the Neighbors Informations List and the EID is deleted from EIDs with unknown MACs List.

4.2.2.6 SSI Reader

The SSI Reader module has the function to obtain the measured SSI from neighbor's packets and update Neighbors Information List accordingly. To obtain the SSI values, this thread accesses directly the Radiotap Header (Figure 2.25 from section 2.4.4) by capturing all wireless frames in the media with a virtual interface working in monitor mode, using libpcap (section 2.4.5.2). As stated in section 2.4.4, one of the informations present in the Radiotap header is the SSI of the received packets.



Configure *pcap* with monitor interface
[finish]
Close *pcap*
Sleep SECURE_TIME
Read one frame
NULL frame
[local node is tracking frame's source MAC]
Read radiotap's SSI and update SSI in neighbors list

Figure 4.13: QCR: SSI Reader Flow Chart

The SSI Reader flow is presented in Figure 4.13 where the thread after initialization process configures the module pcap and associates it to the monitor interface to capture all frames in the media. If the received frame is `NULL` (i.e. frame not received), the thread returns to verify the finish condition. Otherwise, the thread reads the source's MAC address and it verifies if this MAC address is presented in the Neighbors Informations List. If it is true, the thread reads the associated Radiotap Header and gather the SSI and then it updates the SSI value in the Neighbors Informations List.

4.2.2.7 IPC Socket

The Inter Process Communication (IPC) Socket uses a binded UDP Socket (always listening) that uses the loopback interface to allow the local user's applications to communicate with QCR in order to make requests or to give instructions, using this socket. It is necessary that local applications know the communication protocol to communicate with the QCR. The communication protocol consists in the use of the defined packets with the format presented in Figure 4.14.

There is a structure for packets that applications send to the QCR module, called `Instruction/Request general packet`. The first byte contains the command, i.e. the purpose of the packet, and after that, it may appear one or two fields with variable length according to the command. The list of existent commands is presented in Instruction/Request Packets Format which contains also the fields of each type of packet. The existent commands allow an application to: Add a new neighbor by EID; Add a new neighbor by EID which MAC is previously known; Delete an EID and associated information; Delete a MAC interface and associated information; Request the most recent SN by EID or by MAC; Request the SSI by EID or by MAC; Request Quality by EID or by MAC; Delete all EIDs and associated information; verify if the QCR is connected in the respective port or even to finish the application.

For QCR answers, it is used the same socket but the packet's format is different. The `Answer general Packet` is composed by a first byte that indicates an acknowledgment or error and after appears a command, that usually is the received command (those one that QCR is answering). Then, may appear some data if the answer is an acknowledge that needs to send data to the application. The Answer packet formats for each type of answer is presented also in Figure 4.14. The `ERROR` packet is used, for example, when the application request data that is not available (`COMMAND` field is the received command) or when the received packet does not follow the established communication rules (`COMMAND` is `UNK_COMM`).

The IPC Socket flow is presented in Figure 4.15 where at the beginning it is created and binded a socket to allow the communication between the processes, and then the thread is always waiting a new connection with a command. When it receives a command, it reads the command and the thread performs the adequate action. For example, for an `ADD_EID` command, the thread will try to add the packet's content (EID) to the unknown MACs List, and then it will answer to the application with an `ACK` or `ERROR` packet according to the operation success.

4.2.3 QCR Menu Application

To test the QCR module and/or to allow an user to use it with a terminal line, it is available a QCR Menu Application. This application interacts with the user via a terminal and with the QCR via an IPC Socket. The menu is present in Figure 4.16 and it allows the user to: check if the QCR is connected; Add a new EID/MAC association; delete a neighbor by EID or MAC; request the most recent SN by EID or MAC; request the most

Instruction / Request general Packet:

1	(facultative, variable)	(facultative, variable)
COMMAND	Content 1	Content 2

Answer general Packet:

1	1	(facultative, variable)	(facultative, variable)
ACK / ERROR	COMMAND	Content 1	Content 2

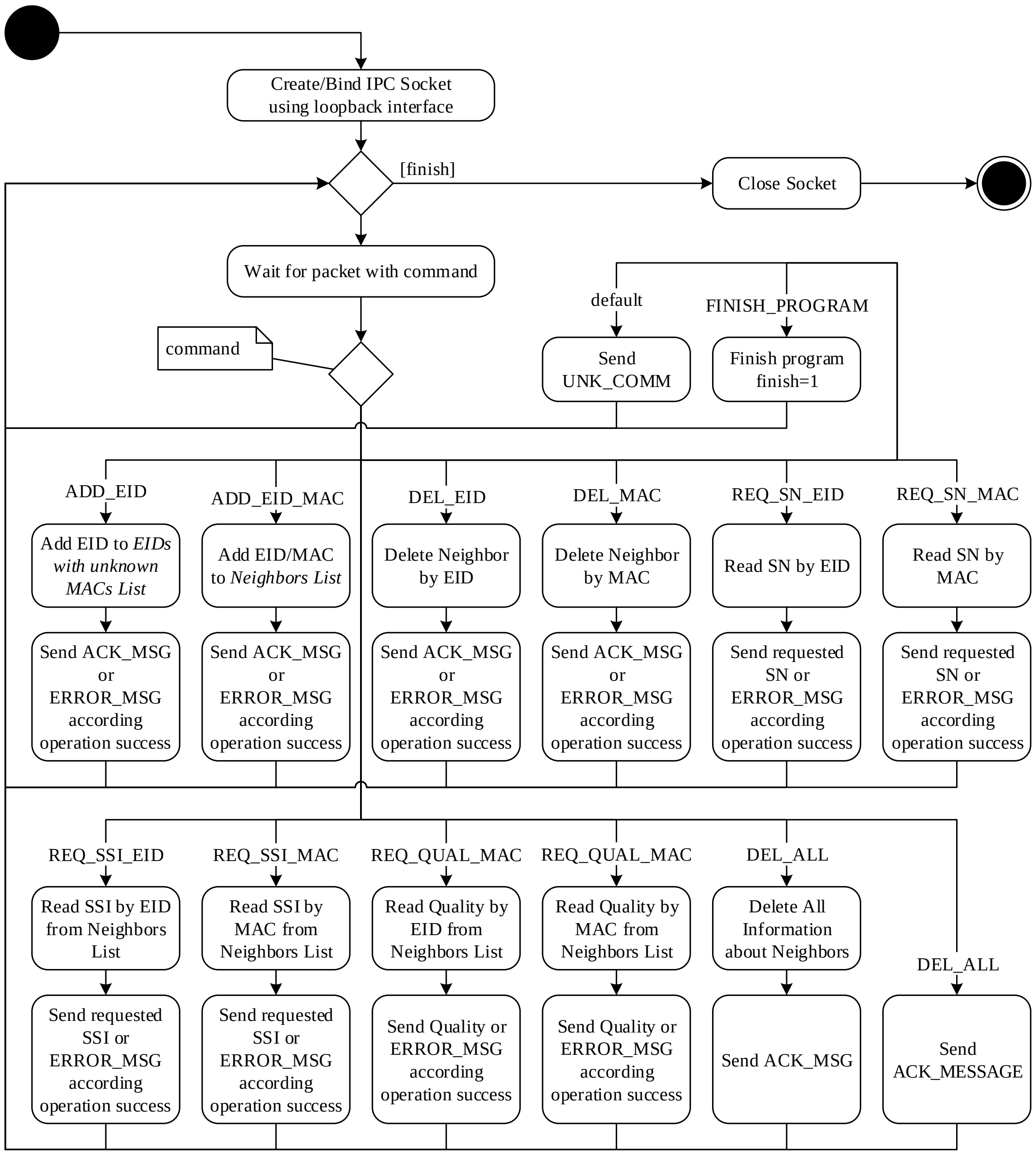
Instruction / Request Packets Format:

COMMAND (1 byte)	Fields (variable)	
ADD_EID	EID	
ADD_EID_MAC	EID	MAC
DEL_EID	EID	
DEL_MAC	MAC	
REQ_SN_EID	EID	
REQ_SN_MAC	MAC	
REQ_SSI_EID	EID	
REQ_SSI_MAC	MAC	
REQ_QUAL_EID	EID	
REQ_QUAL_MAC	MAC	
DEL_ALL		
IS_ALIVE		
FINISH_PROGRAM		

Answer Packets Format:

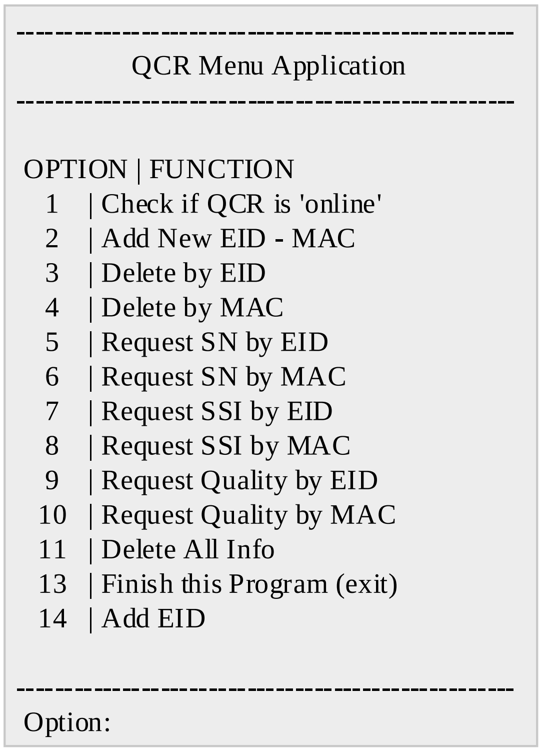
COMMAND (1 byte)	Fields (variable)	
ACK	COMMAND	Content
ERROR	COMMAND	
ERROR (default)	UNK_COMM	

Figure 4.14: QCR: IPC Socket Packets



Create/Bind IPC Socket using loopback interface
[finish]
Close Socket
Wait for packet with command
command
default
FINISH_PROGRAM
Send UNK_COMM
Finish program finish=1
ADD_EID
ADD_EID_MAC
DEL_EID
DEL_MAC
REQ_SN_EID
REQ_SN_MAC
Add EID to EIDs with unknown MACs List
Add EID/MAC to Neighbors List
Delete Neighbor by EID
Delete Neighbor by MAC
Read SN by EID
Read SN by MAC
Send ACK_MSG or ERROR_MSG according operation success
Send requested SN or ERROR_MSG according operation success
REQ_SSI_EID
REQ_SSI_MAC
REQ_QUAL_MAC
REQ_QUAL_MAC
DEL_ALL
Read SSI by EID from Neighbors List
Read SSI by MAC from Neighbors List
Read Quality by EID from Neighbors List
Read Quality by MAC from Neighbors List
Delete All Information about Neighbors
DEL_ALL
Send requested SSI or ERROR_MSG according operation success
Send Quality or ERROR_MSG according operation success
Send ACK_MSG
Send ACK_MESSAGE

Figure 4.15: QCR: IPC Socket Flow Chart



```
--------------------------------------------------
              QCR Menu Application
--------------------------------------------------

 OPTION | FUNCTION
    1   | Check if QCR is 'online'
    2   | Add New EID - MAC
    3   | Delete by EID
    4   | Delete by MAC
    5   | Request SN by EID
    6   | Request SN by MAC
    7   | Request SSI by EID
    8   | Request SSI by MAC
    9   | Request Quality by EID
   10   | Request Quality by MAC
   11   | Delete All Info
   13   | Finish this Program (exit)
   14   | Add EID

--------------------------------------------------
 Option:
```

Figure 4.16: QCR Menu Application Terminal

recent SSI measured by EID or MAC; request the Quality by EID or MAC; delete all neighbors and add a new EID.

The QCR menu is a simple program that connects to the QCR's IPC Socket, according the user's option, to send instructions or to receive data. The QCR Menu Flow Chart is presented in Figure 4.17. In resume the program waits for a user's command, and then it sends the instruction to the QCR. According to the instruction, it waits for the answer that may be an acknowledge/error or data destined to the user.

4.3 Q-PRoPHET: Integration with IBR-DTN

The Q-PRoPHET was implemented in IBR-DTN [29] (presented in section 2.2.6.2), an implementation of DTNs that contains several routing protocols, one of them is the PRoPHET. IBR-DTN was selected due to the reasons presented in section 2.2.6.4, including table 2.1.

To modify the IBR-DTN in order to integrate the new routing protocol Q-PRoPHET, it was necessary to explore the source code of IBR-DTN 1.0.0 with the support of [96]. The modifications performed to integrate the Q-PRoPHET in IBR-DTN are the following:

1. Create a new folder in routing folder named QProphet. Copy the PRoPHET routing source code files to this new folder. Modify these files in order to change classes

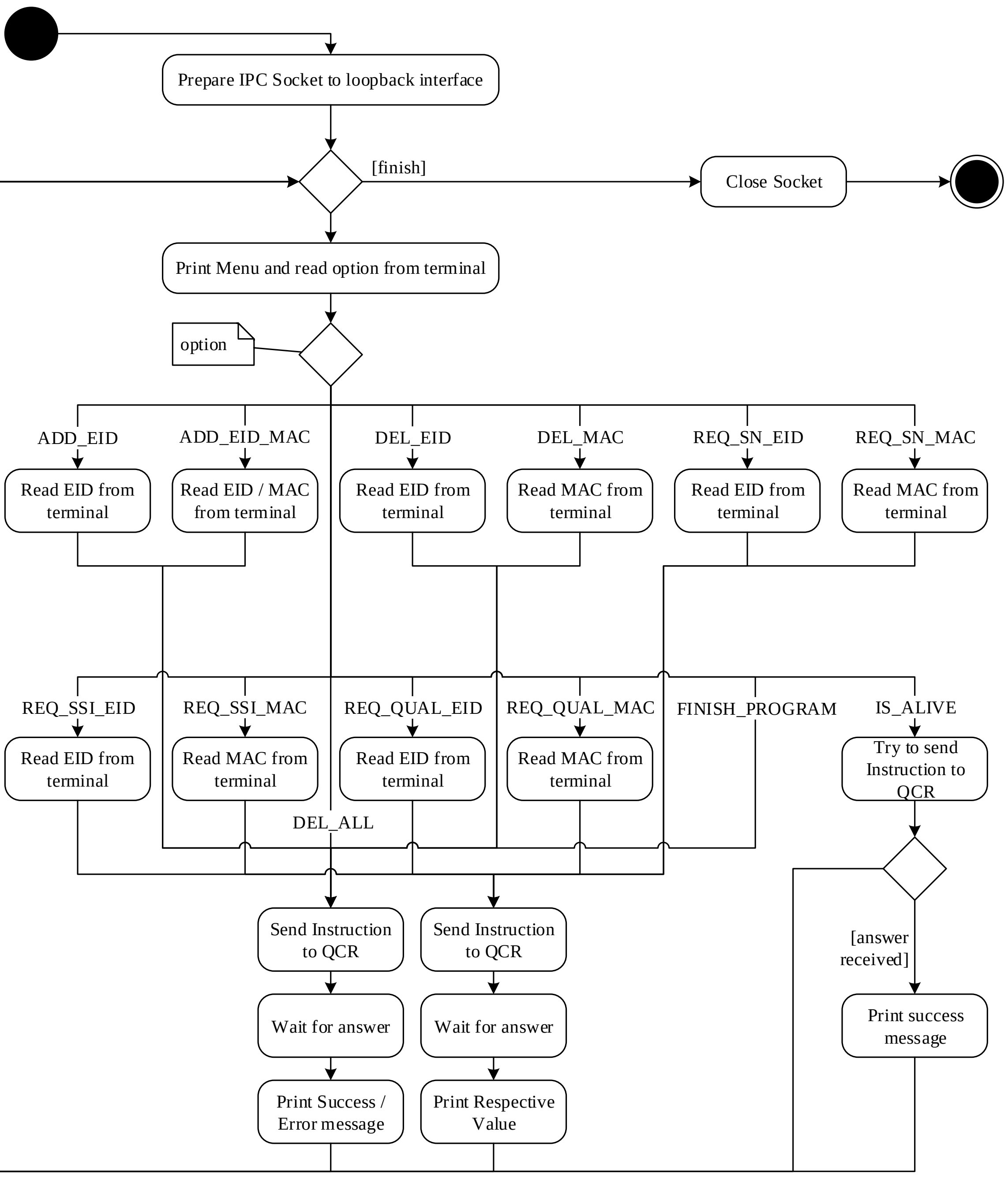


Figure 4.17: QCR Menu Application Flow Chart

names, methods names and attributes names;

2. Set the Q-PRoPHET formulas according to the equations 3.3, 3.4 and 3.5. Set the GRTR and GTMX forwarding strategies according to Q-PRoPHET forwarding strategies presented in section 3.3.2;

3. Create a new class to establish communications with QCR's IPC socket according to the defined protocol (section 4.2.2.7). Change Q-PRoPHET methods to use this new class to communicate with QCR to send or to receive the necessary instructions or informations;

4. Change the *makefiles* in order to allow the compilation and linking phases of all these new code files, when IBR-DTN daemon is compiled;

5. Include the Q-PRoPHET source code files in external files that need to use/access routing classes;

6. Change the IBR-DTN configuration module to insert the support to a new routing protocol, i.e., in order to accept the Q-PRoPHET as a valid routing protocol. It was also added a new class to insert/store the Q-PRoPHET configuration parameters associated to the protocol in IBR-DTN;

7. Change the Management Connection module that may need to access the routing parameters in order to search for Q-PRoPHET extension and to read its parameters or predictability tables;

8. Change Native Daemon module in order to launch the Q-PRoPHET extension when Q-PRoPHET routing protocol is selected in IBR-DTN's configuration file;

9. Insert a new parameter in the configuration file to put the EID of a specific node that is connected via Ethernet and it is considered always available. The Q-PRoPHET considers always this neighbor node has always unitary quality, i.e. the maximum quality possible because the connection is considered perfect.

4.4 Chapter Considerations

This chapter presented the architecture and the implementation of the solution to perform routing in navy scenarios.

First, it presented the architecture and the implementation of QCR, the API responsible to follow and gather the quality of the wireless links.

Then, it presented the Q-PRoPHET architecture and implementation which implements the Q-PRoPHET solution inside IBR-DTN, the selected DTN platform.

The next chapter presents the integration of the implementation of QCR and Q-PRoPHET inside IBR-DTN using a real testbed. It presents the evaluation of these implementations emulating the navy scenarios.

Chapter 5

Integration and Evaluation

5.1 Introduction

This chapter presents the integration of IBR-DTN with Q-PRoPHET and QCR in a real testbed using Single Board Computers (SBCs) with OpenWRT OS, runs experiments in navy emulated scenarios and discusses the obtained results.

Section 5.2 presents a brief description of the hardware and OS of the testbed.

Section 5.3 presents the installation and configuration of IBR-DTN and other modules.

Section 5.4 presents the challenges to emulate the navy scenarios in the testbed.

Section 5.5 presents the evaluation of QCR, and section 5.6 presents the evaluation of Q-PRoPHET in three scenarios that emulate the scenarios presented in section 3.2, two of them are indoor and the last one is outdoor with real mobility of the nodes.

Section 5.7 presents the chapter summary and considerations.

5.2 Hardware and Operating System Description

Along this section it is described the SBCs used to implement and test the solution proposed in chapter 3, as well as to describe the OS installed.

5.2.1 Single Board Computer

The SBCs used in all the evaluation phase are *Cambria Network Computers - GW2358-4* (only Cambria(s) in following sections) (Figure 5.1) developed by Gateworks® [97]. Cambrias were developed mainly for enterprise and residential network applications.

Cambria GW2358-4 [98] main features are:

- Intel® XScale® IXP435 667MHz Processor;
- 128MB SDRAM Memory;
- 32MB Flash Memory;

- Two 10/100 Ethernet ports;
- RS232 Serial Port;
- Real Time Clock;
- OpenWRT Linux Board Support Package.

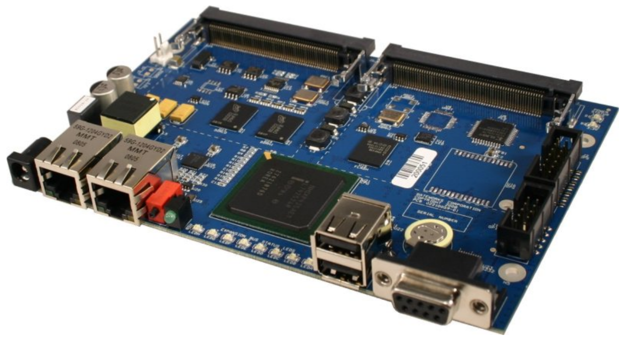


Figure 5.1: Cambria GW2358-4 [97]

All Cambrias were equipped with AR5213A Wi-Fi cards [99], a multi protocol MAC baseband processor from Atheros® Communications which in turn provides support for AG 2.4GHz Wi-Fi.

5.2.2 Operating System

The Operating System installed on Cambrias is OpenWRT Wireless Freedom [100], an open-source GNU/Linux distribution for embedded devices, typically wireless routers. The version is the *Barrier Breaker 14.07* (based on r42625). OpenWRT offers a fully file system with package management in order to enable developers/programmers to customize and configure the system to suit their needs, as well as to develop their own applications. OpenWRT is on constant evolution due to several contributions from OpenWRT community. See reference [100] for more details.

In the user point of view, OpenWRT is similar to Linux where the user interacts with the OS usually via a terminal, or in some cases via a graphical interface (e.g. web page) provided by some software packages. These software packages have the disadvantage of loosing the freedom to perform some tasks, depending on the package in use.

OpenWRT offers an environment to facilitate the developers work. It provides the OpenWRT Buildroot, which is a set of makefiles and patches that may be installed in the user's host system (usually a laptop). Buildroot allows the programmer to easily compile and to create packages to install their programs in OpenWRT OS. It contains a toolchain that allows the programmers to compile their programs to OpenWRT in their personal *host systems*. This process of compilation is called cross-compilation because it allows to compile a program to a processor architecture in a machine containing a different architecture, i.e.

with different Instruction Set Architectures (ISAs). It is possible to perform this process manually using the toolchain, but buildroot automates for a large quantity of embedded systems which facilitates the user's tasks.

5.3 Cambrias Configuration and Software Integration

This section presents the relevant configurations and the steps to install the different software on Cambrias.

The Cambrias used in the evaluations may have special configurations from the scenario, but some configurations are independent. The most relevant and common configurations are presented in this section.

5.3.1 Cambrias Configuration

First it is installed OpenWRT *Barrier Breaker 14.07* OS in all Cambrias. But before, it is installed the OpenWRT Buildroot on a laptop with Linux OS. Then, Buildroot is configured to compile the OpenWRT (and all selected packages) to the target system Intel IXP4xx (Cambria's processor family). Then, the OpenWRT OS and the boot-loader are compiled and installed in all Cambrias.

Second, the basic network configurations are performed. It is created two virtual wireless interfaces (wifi-iface) converging in the same radio interface (wifi-device). One of them is configured in Ad-Hoc mode and the other is configured in Monitor mode. One Ethernet port is configured to obtain automatically IP via a Dynamic Host Configuration Protocol (DHCP) server in order to facilitate the access to the boards by Secure Shell (SSH) inside the building's intra-network when possible.

5.3.2 Software Integration

The software to be integrated in Cambrias is IBR-DTN (modified version with Q-PRoPHET) and the QCR module.

The IBR-DTN (version 1.0.0) modules were compiled in Buildroot and installed in all Cambrias in the following order: *dtndht*, *ibrcommon*, *ibrdtn*, *ibrdtnd* and *ibrdtn-tools*, where ibrdtnd was previously modified in order to include the Q-PRoPHET.

The QCR module was compiled using directly the toolchain because it facilitates the process of development and compilation, from the beginning, in the *Eclipse Kepler* IDE. Then it was generated the executable file and it was installed in all Cambrias; it runs in the OpenWRT from the beginning of the system as an internal service available for local applications to use.

5.4 Evaluation Challenges

The evaluation phase presented several challenges and constrains that forced to change and adapt the planned procedures to perform the implementation and its evaluation. The most important constrains are described along this section.

The first constrain was due to libpcap performance to capture IEEE 802.11 beacon frames in monitor mode, because its performance varies significantly according to the platform being used [92]. The Cambrias have limited capabilities, a 667MHz processor and 32MB of memory. This was not enough for the board to capture and process all existent frames at an indoor environment, and this affected the QCR sensitivity to measure the quality of the relevant links. However, in an outdoor environment it was possible to capture more relevant frames than in the inside which increased the QCR's sensitivity to measure the quality. Another constrain that affects the beacons transmission was the fact that, if a certain node is almost to send a beacon and it receives a beacon from another node, it waits some time until it tries to send the beacon again. Due to the large amount of beacons on the media, it may increase and delay the sending phases of the beacons. The implementation of QCR's beacon system solves this problem and it allows more freedom in the beacon management, i.e. it is easy to change the beacon periodicity, for example. However, if a node wants to track another node, both of them need to run the QCR, because is it the one that sends the beacons.

Another problem that happened with the QCR module is the fact that the high amount of analyzed frames required high resource consumption which increased the CPU usage and consequently the system's load average[1]. Due to this constrain, it was inserted a delay phase between frame captures, which decreases the number of analyzed frames. Consecutively, not all frames are captured, but it is not necessary to listen to all frames. The miss of some frames from DTN neighbors is not a problem, since they are only necessary to get the SSI values; the beacons system is implemented in the transport layer by QCR. The delay phase in QCR's SSI Reader thread allows to solve the majority of the load problems that occurred in the QCR module.

However, IBR-DTN evaluation also presents load problems. Despite the fact that IBR-DTN has been developed for systems with low CPU and memory capabilities, it presents load problems to run in Cambrias when it needs to perform its tasks: if the IBR-DTN is running without tasks to perform (e.g. send or receive bundles), it does not present problems; but if it is necessary to send or to receive files, the load increases and affects the system's performance. This problem limits the freedom to perform tests in Q-PRoPHET due to the fact that, if there is a large amount of files or large files to transfer, the load values increase and the results may not be valid. To minimize these problems, methodologies were proposed to perform the tests that will be more detailed in the relevant sections.

Another big challenge was to emulate the proposed scenarios for communication to

[1]The load average measures the mean of the number of processes waiting to be served by the CPU. This number should be lower than the total number of CPUs because that means that the CPUs can dispatch all processes. For a single CPU system, the load should not be higher than 1 ideally. Unix-like systems usually display the load information relative to the last 1, 5 and 15 minutes [101].

test the Q-PRoPHET. It was necessary to create/emulate links with good communication and links with bad connections, i.e. intermittent links. However, positioning the boards in specific configurations in order to emulate intermittent conditions is a difficult task to perform due to the wireless medium instability. To try to solve this problem, scripts have been deployed to vary the antennas transmission power. However, the variation was not enough to emulate all scenarios; the first scenarios have been emulated this way at the building's corridors. To solve these problems in the second scenario, it was performed tests with Cambrias near each other without antennas. This way, it was created a "controlled" environment with the desired intermittence conditions.

To test the different scenarios, it was necessary to measure the times in different boards which implies to have the Cambrias synchronized between them. First, it was used the IBR-DTN synchronization system, but this system was not effective to synchronize in an efficient way all boards and some measures of e2e delay were negative. Thus, to perform the indoor tests and taking into account that all boards are connected in the same intranet with Internet access, it was used NTP [19] [102] to synchronize all boards using an NTP server present in the Internet. However, to test the outdoor scenario, it was not possible to connect all boards to the Internet to provide synchronization between all boards. So, it was used one fixed Cambria with Internet access to synchronize itself by NTP and providing its own clock using PTPd [20].

Finally, the IBR-DTN was not developed to perform routing in the conditions that Q-PRoPHET requires, where it is necessary to verify always the available neighbors, instead of generating events only when a new neighbor appears or disappears, i.e. it is discovered by the Discover module or it disappears the necessary amount of time for the Discover module to realize that it disappears. This fact limits its capability to obtain always the more recent quality values and to maintain a consistent table of neighbors. Ideally, it should always poll the quality value, which certainly increases the load average of the system, that may be a problem depending on the host system.

5.5 Quality Connection Reader

To test the quality values measured by QCR, two Cambrias are near each other by a distance of `d` as presented in Figure 5.2 in an outdoor environment.

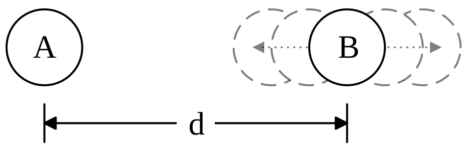


Figure 5.2: Scenario to Measure the Quality Factor

To perform this test, node A is static and node B can be mobile. In the experiment, it is measured the quality factor between nodes A and B, as well as the SSI and the Stability parameters between 2m and 100m, and capturing 200 beacons frames (nearly 40 seconds) at each measured distance. With the different values of quality, SSI and LStab measured, the obtained results in this test are presented in Figure 5.3 with a confidence interval of 95%.

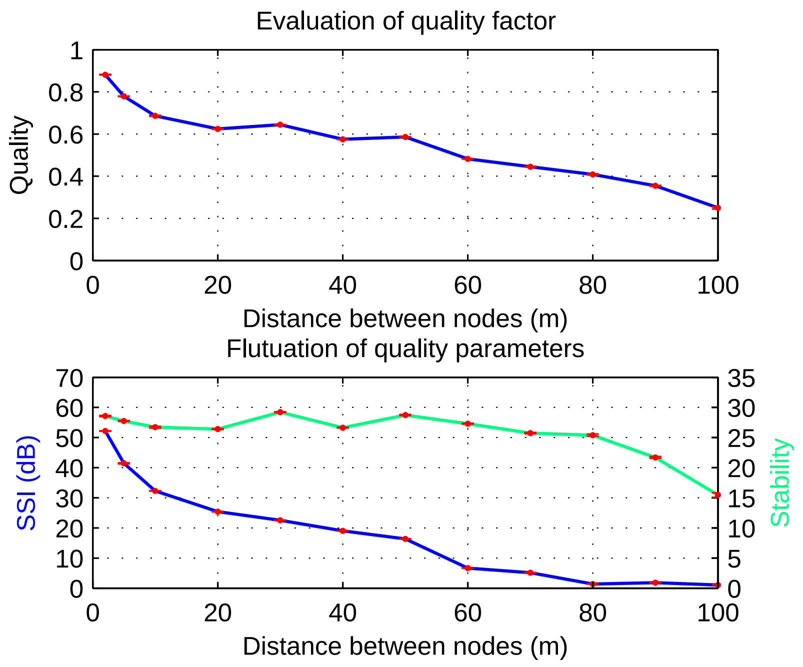


Figure 5.3: Evaluation of Quality Factor and Quality Parameters

The obtained results in this scenario show that the quality factor decreases with the distance, as expected, due to the fact that the SSI and the Stability are decreasing with the distance. This result was expected and it was the desired result in order to quantify the quality of the links.

However, there is a behavior that is important to refer. The SSI decreases more for lower distances, with no significant differences at higher distances where the communication is equally possible, because the stability is still high. Instead, the stability for lower distances does not vary significantly, and it begins to decrease only for long distances (>60m). Thus, the quality has an approximately decreasing behavior due to the behavior of the quality parameters (SSI and LStab). This way, it is possible to verify that the SSI is not the best metric to evaluate the link quality, because at higher distances it loses sensitivity and cannot define well the communication. At high distances, the stability has better sensitivity. Using these two parameters, the quality factor can have a good sensitivity at all distance ranges.

The QCR module is capable to describe and define well the quality of the connections.

5.6 Quality-PRoPHET

To evaluate the Q-PRoPHET routing protocol, implemented into IBR-DTN, the scenarios for communication presented in section 3.2 were emulated, and the results obtained for Q-PRoPHET were compared with the results obtained for PRoPHET evaluated in the same conditions.

Both protocols contain parameters and configurations that affect the way they work. The main values set in the configuration file that affect the routing modules are presented in Table 5.1.

Table 5.1: Definition of routing parameters in IBR-DTN configuration file

Parameter	**PRoPHET**	**Q-PRoPHET**
routing	prophet	QProphet
routing_forwarding	yes	yes
routing_prefer_direct	no	no
p_encounter_max	0.7	(not applicable)
p_encounter_first	0.5	(not applicable)
p_first_threshold	0.1	0
beta	0.9	0.9
next_exchange_timeout	10	10
forwarding_strategy	GRTR	GRTR
gamma	0.999	0.999

The other important configurations set in IBR-DTN configuration file are presented in Table 5.2. Note that the fragmentation is disabled: the objective is to evaluate the routing protocol in different scenarios, but also with different sizes of bundles which cannot be controlled with the fragmentation enabled.

Table 5.2: Definition of general parameters in IBR-DTN configuration file

Parameter	**Value**
fragmentation	no
limit_payload	10MB
limit_storage	1GB
net_interfaces	lan0
net_lan0_type	tcp
net_lan0_interface	ath0
net_lan0_port	4556
time_reference	yes
time_synchronize	no
time_discovery_announcements	no
time_set_clock	no

These are the default parameters used to perform the evaluation in the scenarios for communication. Any parameter different from the defined parameters is described in the respective section. If not, the default values are used.

5.6.1 Scripts and IBR-DTN source code modifications

5.6.1.1 IBR-DTN modifications to gather data logs

It was necessary to perform changes in IBR-DTN source code in order to generate the necessary data logs to track the departure and arrival times associated to relevant bundles in order to verify which bundles are delivered and to measure the e2e delay between source and destination nodes. To obtain these informations the following changes were performed:

- **BundleReceivedEvent:** The `getMessage` method was changed in order to filter the bundles associated to the application `dtnSender` and write in a specific file the following information about the received bundle separated by commas:
  - **host_id:** The host that generated the information log;
  - **src_id:** The source of the bundle;
  - **dest_id:** The destination of the bundle;
  - **timestamp:** The bundle's origin timestamp in seconds after year 2000;
  - **seqNum:** The sequence number of the bundle;
  - **time:** The time in seconds (and microseconds) after year 1970 which the information log was generated.
- **TransferCompletedEvent:** The `getMessage` method was changed in order to filter the bundles associated to the application `dtnSender` and write in a specific file, different from the previous one. The following information about the bundle is sent separated by commas:
  - **host_id:** The host that generated the information log;
  - **src_id:** The source of the bundle;
  - **dest_id:** The destination of the bundle;
  - **timestamp:** The bundle's origin timestamp in seconds after year 2000;
  - **seqNum:** The sequence number of the bundle;
  - **time:** The time in seconds (and microseconds) after year 1970 which the information log was generated.

With these modifications it is possible to uniquely identify all bundles sent by the application identified by `dtnSender`, and store the relevant log information to discover which bundles are delivered and how long they were delivered.

5.6.1.2 Scripts to generate data logs

CPU usage and load average

Due to the CPU usage and load consumption problems stated before, it was necessary to develop a script to measure the CPU consumption and the load average in order verify if the obtained results from the tests are valid or if it is necessary to adapt the test in order to consume less resources from the boards.

The flowchart presented in Figure 5.4 represents the process of the measurement of CPU consumption and load average, as well as the process to store the information in the log files. The information associated to the CPU is obtained using the information present in `/proc/stat` and the information about the load average is obtained from the file `/proc/loadavg`. The gathered information is stored in two different log files where one contains the values of CPU usage and the other contains the values of load average. The \$1 is the input argument and represents the measurement periodicity in seconds, i.e. the time between consecutive measures.

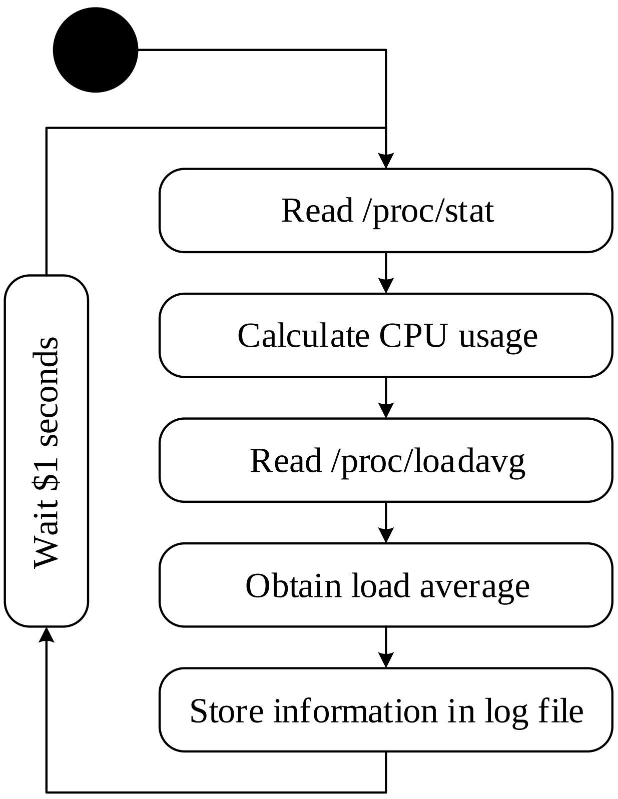


Figure 5.4: CPU/Load Measure Flow Chart

Send and receive files

To perform the different tests, it is necessary to send files from the source node and receive these files in the destination node. To perform this task automatically in the source and destination node, it was created a script to run in these nodes to launch the sender and the receiver applications.

The script accepts four arguments: the number of files to send, the size of each file, the destination node and the time interval between files. Figure 5.5 presents the flow chart of the script.

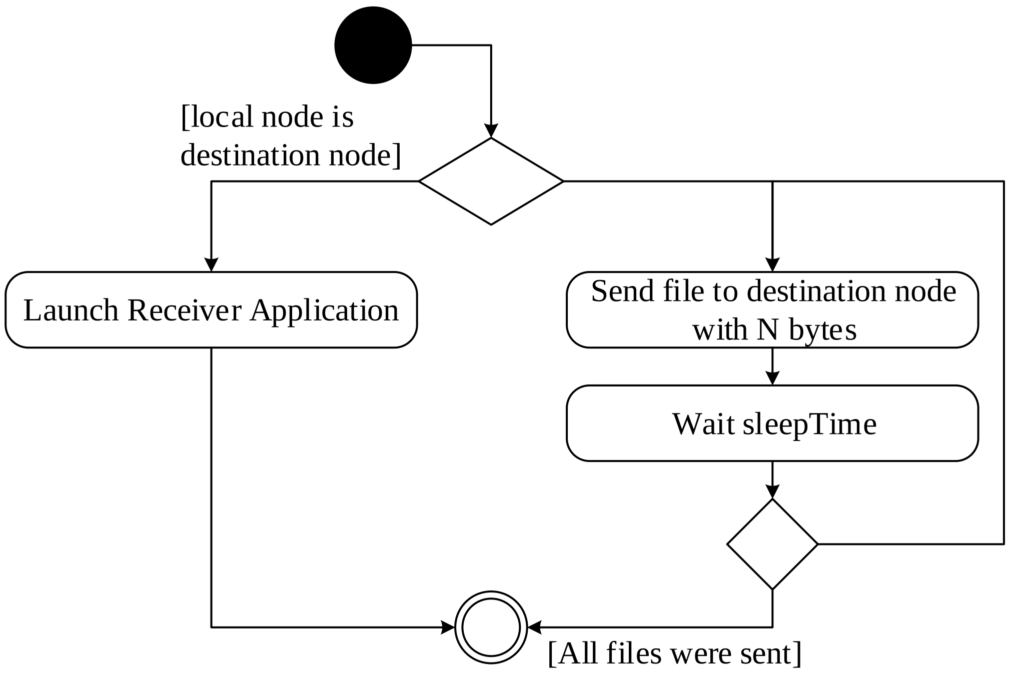

[local node is destination node]
Launch Receiver Application
Send file to destination node with N bytes
Wait sleepTime
[All files were sent]

Figure 5.5: Send/Receive Files Script Flow Chart

If the local node is the destination node, the `dtnrecv` tool is launched with the application name of `dtnReceiver` and the script finishes. If not, the script initiates the phase to send files with `N` bytes to the destination node with intervals of `sleepTime`. To send each file is used `dtnsend` tool with the application name of `dtnSender`. When all files are sent the script finishes.

5.6.1.3 Scripts to perform the tests

To perform the evaluation phase of scenario 1 (Inspection/Boarding) and scenario 2 (Naval and Amphibious Operations) it was developed a script to run in a computer (Master Script) that is responsible to access and control all boards in order to make the tests and repeat them in order to have several repetitions of the same experiment. Figure 5.6 presents the Master Script Flow Chart.

The Master Script accepts arguments such as: the number of iterations per protocol, the experiment time, the size of each file and the number of files to sent, and optionally the intermittence period if it is to use intermittence in the experiment.

First, the script begins by creating a folder with the current date and hour to uniquely identify each experiment, and then it creates a new folder to include the results of each iteration.

Then, the protocol is set to Q-PRoPHET and it is created a folder to include the log files about this protocol.

After, the old logs in Cambrias are deleted if they exist, and then the script enters in a waiting phase until the load average (1 minute) is lower than 0.2, in order to guarantee that all boards in the beginning of the experiment are in the conditions to perform the test and guarantee that this test is not affected due to the high load average in the boards.

If the experiment that is running considers the use of the intermittence script in the boards, the intermittence is enabled. Thus, the intermittence script will be run in the specified boards to vary the antennas' transmission power between the minimum value (1dBm) and the maximum value (16dBm) to simulate intermittent conditions.

Thus, the Send/Receive Files Scripts are called in the Source and Destination nodes, and IBR-DTN is launched in the remaining boards in order to begin the experiment; the Master Script waits the experiment time until it finishes the experiment by finishing all scrips and applications called in the process.

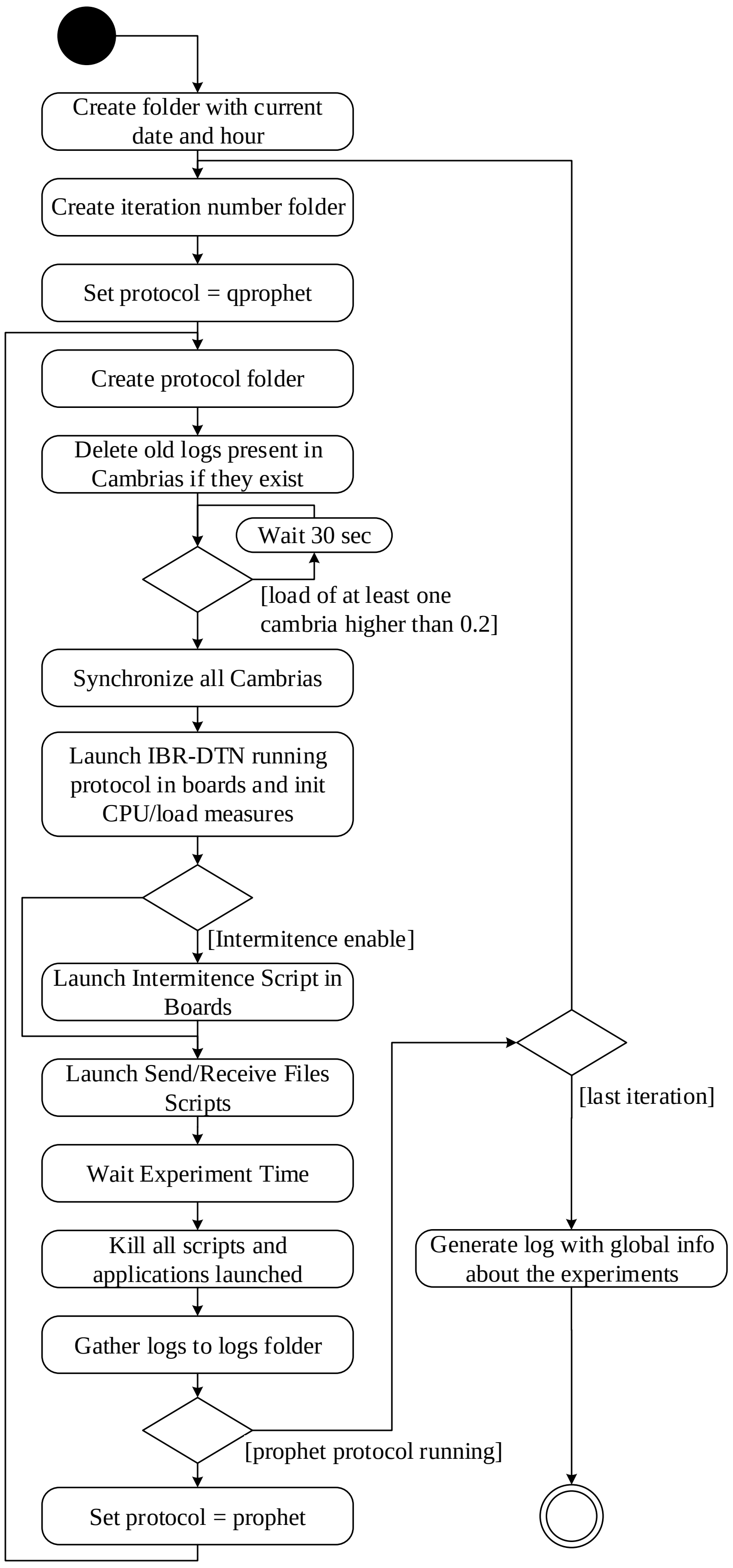
To finish the actual iteration the Cambrias' logs are copied to the respective protocol folder inside the respective iteration folder. Then, the protocol is changed and the experiment is repeated again with the new protocol. Each pair of experiments, one from PRoPHET and another from Q-PRoPHET, originates the logs from one iteration. Then, if it is the last iteration, it is created a log with general information about the experiments such as: the actual date, the boards identification, the source node, the destination node, the number of files sent per experiment, the size of each file, the number of iterations and the experiment time. Thus, the script finishes and the experiments are done.

5.6.2 Scenario 1: Inspection/Boarding

5.6.2.1 Scenario Definition and Emulation

The Inspection/Boarding scenario is emulated with three nodes as presented in Figure 5.7. In this scenario the objective is to create a stable connection between nodes A and B, and also between nodes B and C. However, it is desirable to create an intermittent connection between nodes A and C to emulate intermittent conditions between these two nodes to verify the behavior of both protocols under this intermittent connection.

The scenario from Figure 5.7 is emulated using three Cambrias inside the building 2 from Instituto de Telecomunicações in Aveiro. To emulate the scenario with the communication conditions stated before, it is necessary to use the *ping* tool in order to guarantee a



Create folder with current date and hour
Create iteration number folder
Set protocol = qprophet
Create protocol folder
Delete old logs present in Cambrias if they exist
Wait 30 sec
[load of at least one cambria higher than 0.2]
Synchronize all Cambrias
Launch IBR-DTN running protocol in boards and init CPU/load measures
[Intermitence enable]
Launch Intermitence Script in Boards
Launch Send/Receive Files Scripts
Wait Experiment Time
Kill all scripts and applications launched
Gather logs to logs folder
[prophet protocol running]
Set protocol = prophet
[last iteration]
Generate log with global info about the experiments

Figure 5.6: Master Script Flow Chart

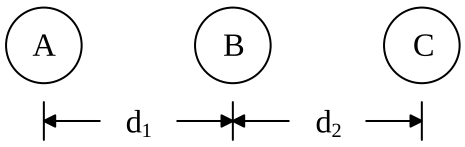


Figure 5.7: Inspection/Boarding Scenario Emulation

full communication between node B with its neighbors, and in order to discover the limit of communication between nodes A and C. These limits are discovered for a transmission power (txpower) of 16dBm in all antennas. Then, the txpower is varied to 1dBm in nodes A and C to verify that the connection between these nodes is really bad and the communication between node B and the other nodes is stable and good. The obtained board's disposition inside the building is presented in Figure 5.8. This way it is possible to control the communication between nodes A and C by controlling the board's transmission power.

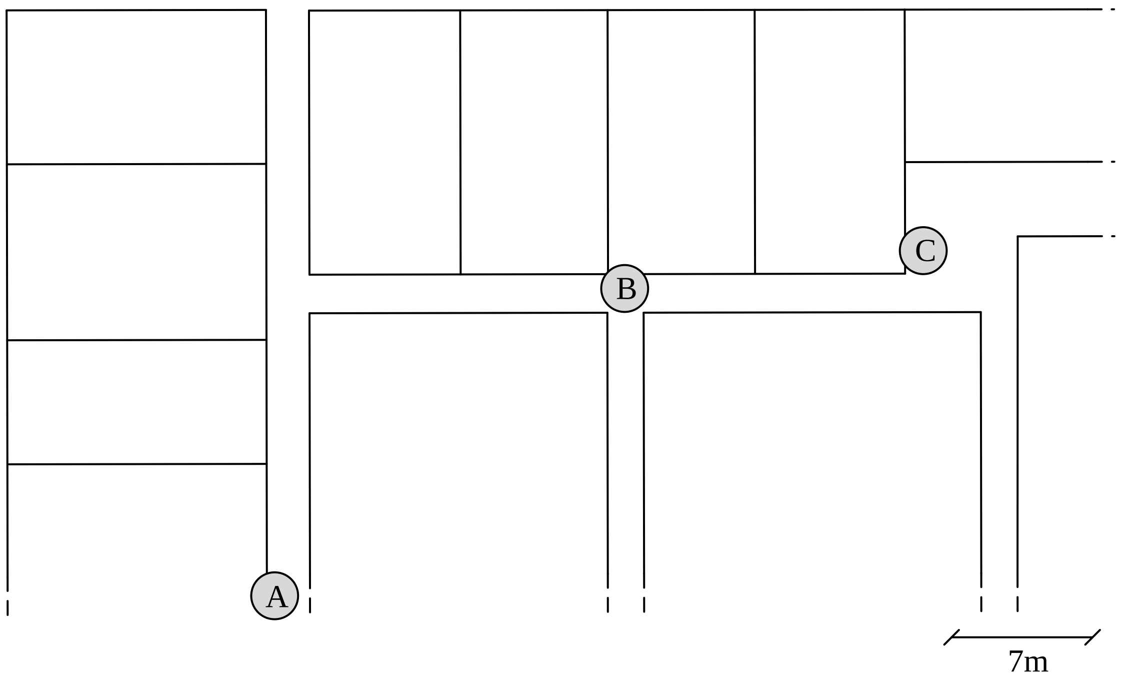


Figure 5.8: Inspection/Boarding Boards Disposition

5.6.2.2 Evaluation Procedure

Three experiments are performed with different file sizes: 1KB, 15KB and 85KB. Each experiment tests PRoPHET and Q-PRoPHET protocols with 50 iterations for each experiment. Per experiment, it is sent 25 files with a delay of 4 seconds between consecutive files. This way, it is sent 25 files in the first 100 seconds from the experiment to distribute the CPU consumption to send the files along the experiment time. The time of each experiment is 105 seconds and then the experiment finishes and the data is stored. The source and destination boards also run the intermittence script with an interval of 6 seconds, i.e. during 6 seconds the transmission power is 16dBm, and the next 6 seconds the transmission power is 1dBm and so on.

To perform the evaluation in this scenario it is used the script described in section 5.6.1.3. All Cambrias are connected in Instituto de Telecomunicações' intranetwork and the laptop that runs the Master Script is also connected to the same network. This way, the script is able to control all boards and run the experiments automatically.

5.6.2.3 Obtained Results

The delivery ratio results are presented in Figure 5.9 with a confidence interval of 95% for 25 runs of each experiment. It is possible to verify that for a small size of the files, i.e. 1KB and 15KB, there are no significant differences in the delivery ratio, but it is important to refer that Q-PRoPHET is able to deliver 100% of the sent files. For 85KB files there are significant differences in PRoPHET performance because its delivery ratio decreases to 80%, unlike the Q-PRoPHET that continues nearly 100%. This result makes sense because it is expectable that the source node running PRoPHET protocol sees many times the destination node and tries to send the bundles directly. However, it is not possible to send bundles at all times and the source node considers itself a better node to deliver the bundles than the middle node; in Q-PRoPHET the middle node is always a better node to deliver the bundles to the destination.

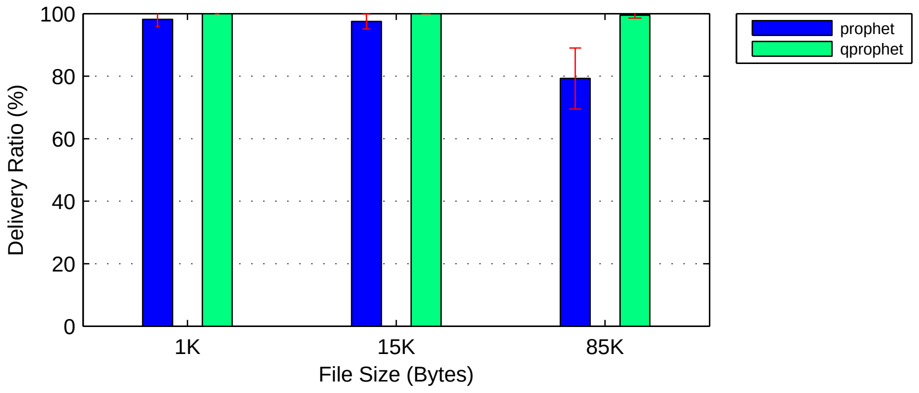


Figure 5.9: Scenario 1: Delivery Ratio

The e2e delay results are presented in Figure 5.10 with a confidence interval of 95%. The e2e delay is measured as the mean time to deliver the packets while a periodic intermittency of 6 seconds is introduced in the source and destination nodes. It is possible to verify that, for all sizes of files, the Q-PRoPHET provides faster delivery than PRoPHET, since it chooses the best-quality available path. Note that, for 85KB files, the delay difference is significant, where Q-PRoPHET spends approximately 8 seconds less to deliver all bundles when compared to PRoPHET.

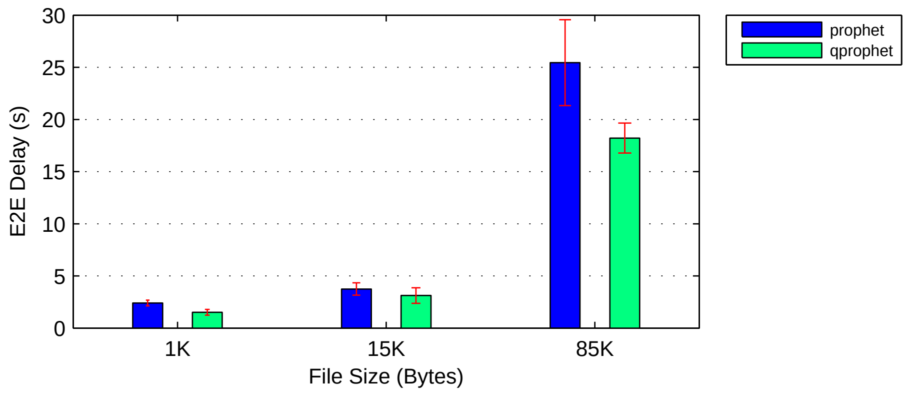


Figure 5.10: Scenario 1: e2e delay

The measurements of CPU usage and load average are presented in Figure 5.11 and Figure 5.12, respectively. These measurements are important to evaluate if the previous results are valid or not, i.e. if the boards are overloaded (high CPU consumption and load average) the results are not valid. In the experiments with 1KB and 15KB files, the CPU usage is often under 50% and the load is near 0.5, which are acceptable values to these parameters. In the experiment with files of 85KB, the CPU usage and load average are higher than in the previous experiments. However, due to the fact that the load average does not cross much the limit of value 1, the results are also considered valid.

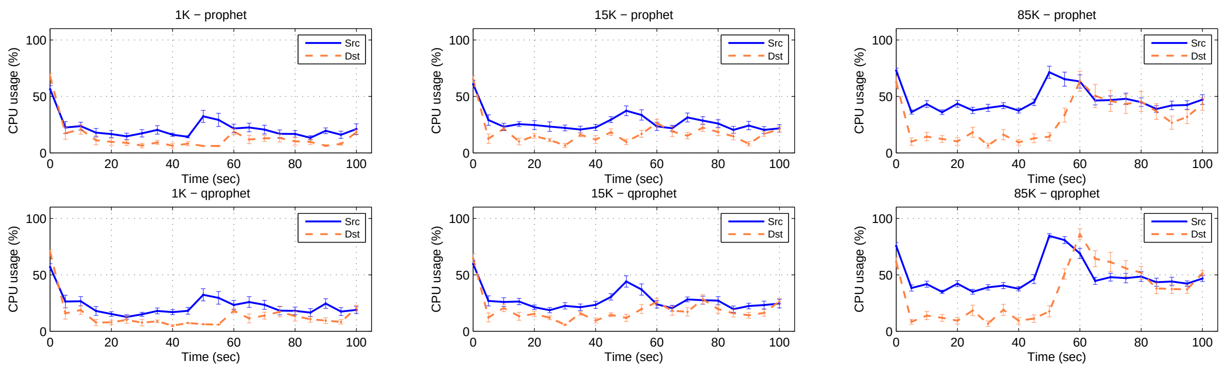


Figure 5.11: Scenario 1: CPU usage

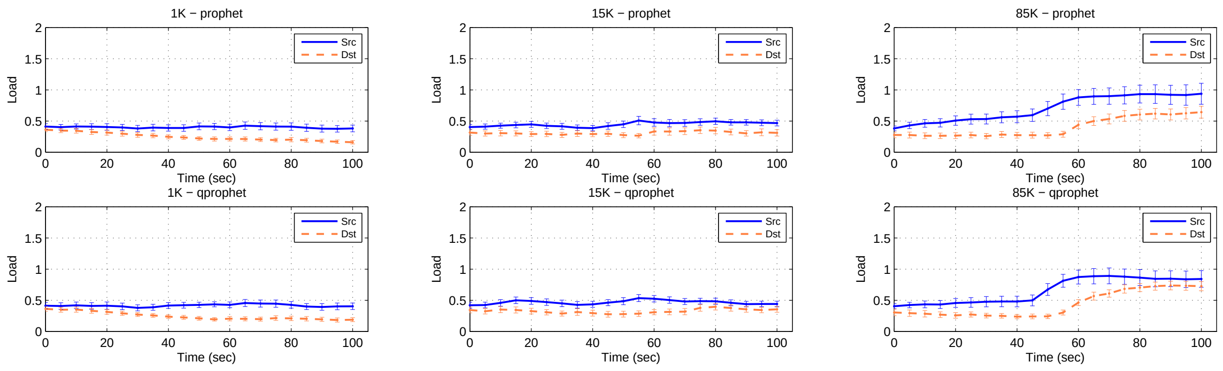


Figure 5.12: Scenario 1: Load average

5.6.3 Scenario 2: Naval and Amphibious

5.6.3.1 Scenario Definition and Emulation

The Naval and Amphibious scenario is emulated using five nodes as presented in Figure 5.13. In this scenario the objective is to send data from node A to node E where these two nodes have not direct contact between each other. Thus, the objective is to use always the middle nodes (not necessarily all nodes) to deliver the information. Due to the fact that the scenario is characterized by some intermittent connections, the boards are placed in a specific disposition where the conditions of the links are checked using the QCR Menu to evaluate the quality of the links.

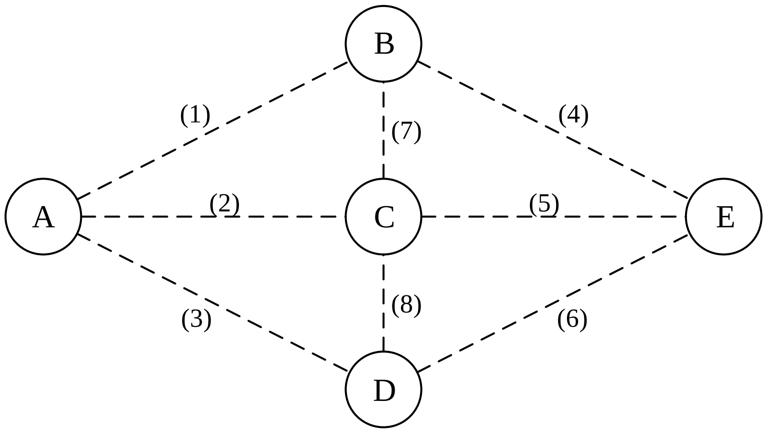


Figure 5.13: Naval and Amphibious Scenario Emulation

Initially, the idea was to place the boards inside of Instituto de Telecomunicações, but it was very difficult to create good and bad links to emulate the scenario because the links never maintain their state, i.e. during some time the link presents the wanted behavior, but then the behavior changes. For example, the creation of an intermittent link was achieved, but after some minutes the link was completely lost.

Due to all challenges to create intermittent conditions, the scenario was emulated with no antennas in the boards and all boards were placed in a table. After several changes in the boards disposition, the final measurements of ping successful and quality values for each link are presented in Table 5.3. The quality values presented in the table are obtained with the mean of two measures of the quality value, each measure from one node to another node and then in the inverse way. It is possible to verify that there are a set of good links which are links (1), (2), (3), (7) and (8); there is one bad link which is link (6) and some intermittent links which are links (4) and (5).

Table 5.3: Scenario 2: Ping Success and Quality of the Links

Link	**Ping success /100**	**Quality mean**
1	91	0.373592
2	100	0.514792
3	99	0.565521
4	83	0.325456
5	72	0.248232
6	0	0
7	100	0.572710
8	99	0.547188

5.6.3.2 Evaluation Procedure

To perform the evaluation in this scenario it is also used the script presented in section 5.6.1.3 with the intermittence script disabled, due to the fact that the connections are emulated in a more controlled scenario than in scenario 1 and presenting natural intermittence.

To evaluate scenario 2 experiments are performed for the following size of files: 1KB, 5KB, 15KB, 35KB and 85KB. To each size of files, PRoPHET and Q-PRoPHET are evaluated, with 25 iterations to evaluate the performance of the routing protocols with a confidence interval of 95%. 20 files are sent from node A to node E with an interval between files of 5 seconds, i.e. all files are sent during the first 100 seconds.

During the experiments it is obtained the necessary data to calculate the delivery ratio, the e2e delay, the CPU usage/load average and the total number of bundles sent in the network.

5.6.3.3 Obtained Results

The delivery ratio results are presented in Figure 5.14. It is possible to verify that for small files, the delivery ratio results are always near 100% for both protocols under test. The main difference is visible for bundles of 85KB, either in PRoPHET and Q-PRoPHET.

In PRoPHET the delivery ratio is approximately 70%, and in Q-PRoPHET it is approximately 90% - 95%, relatively better than PRoPHET.

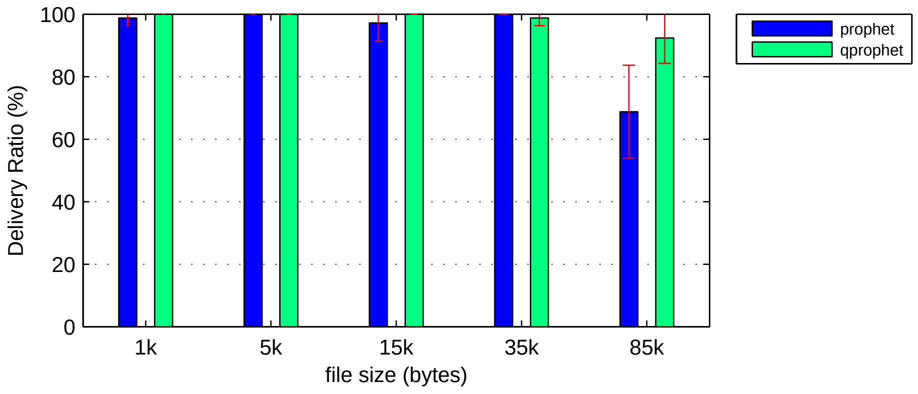


Figure 5.14: Scenario 2: Delivery Ratio

The e2e delay results are presented in Figure 5.15. It is possible to verify that there is a trend of higher e2e delays with the increasing of the files size. However, there is an anomaly in the tendency in PRoPHET for files of 35KB. For all sizes the e2e delays in Q-PRoPHET are lower (or equal for 35KB) than the delays in PRoPHET protocol. In this scenario Q-PRoPHET will make use of the available best quality links and will be able to send the same information in less time. Note that, the e2e results obtained for this scenario are higher than the e2e results obtained for scenario 1. It means that the number of intermittent links and its effect in the e2e results have impact in both protocols.

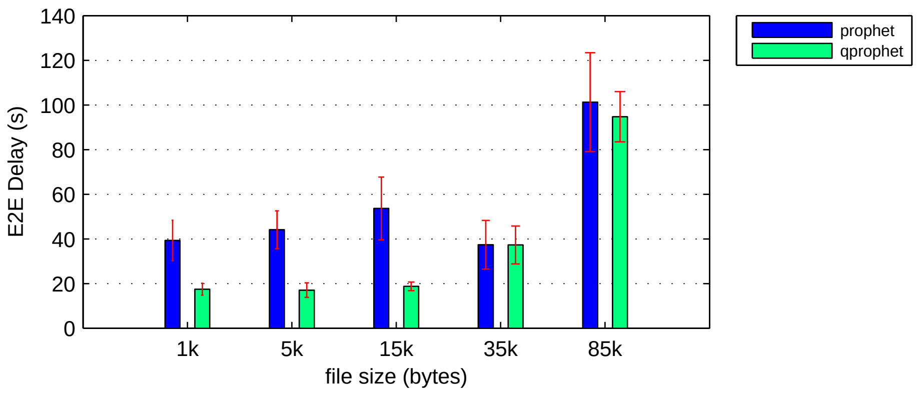


Figure 5.15: Scenario 2: e2e delay

The total number of transmissions in the network, i.e., in all nodes, can be seen in Figure 5.16. It is possible to verify that Q-PRoPHET always transmits more than PRoPHET. Higher packet transmissions means that Q-PRoPHET has better opportunities to transmit packets, with more available quality links to the destination.

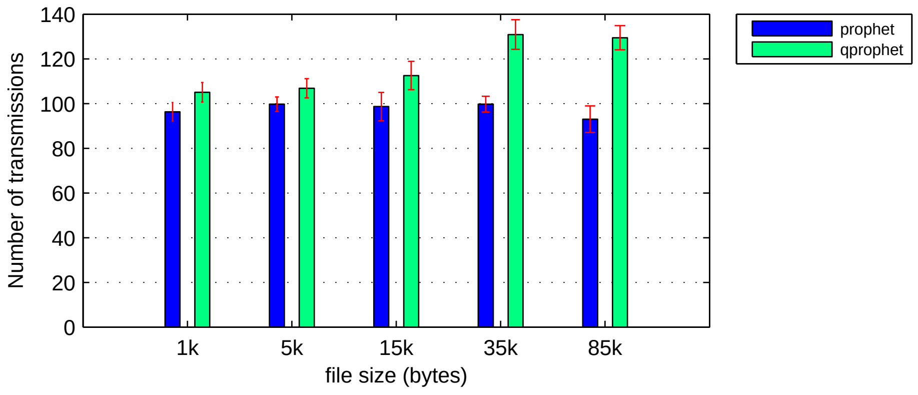


Figure 5.16: Scenario 2: Total transmissions per Experiment

The measurements of CPU usage and load average (1 minute) are presented in Figures 5.17 and 5.18, respectively. For files of 1KB to 35KB, the load average never reaches value 1 for both protocols, which means that the boards deal well with the experiments. For the experiments with 85KB files, the load crossed value 1, but it does not diverge and never reaches value 2. We then consider that the results are valid.

5.6.4 Scenario 3: Population Support

5.6.4.1 Scenario Definition and Emulation

The Population Support scenario is emulated in an outdoor environment and using real mobility. To emulate the scenario, two static boards (D and E) are placed in the two buildings of Instituto de Telecomunicações in Aveiro, one board at each building, as presented in Figure 5.19. These two boards have an Ethernet connection to the buildings' intranetwork, which may be used as a static route always available between these two boards. Note that boards D and E cannot communicate via wireless. There are more three nodes: A, B and C, which will move during the experiment. The initial position of these three nodes is presented in Figure 5.19. Nodes A and B will move in clockwise direction, and node C moves in counterclockwise. Nodes B and C encounter each other at the bottom right corner of Building 2 and in the superior left corner. The mobile nodes delay, approximately, 210 seconds to give one lap.

Nodes A and B may communicate often directly because they are near each other and generally they do not have obstacles between them. Depending on the position, these nodes may communicate directly with nodes D or E or even C. For node C it is applicable the same aproach.

5.6.4.2 Evaluation Procedure

To test this scenario, node C is selected as the source node, and node B is the destination node. Each experiment in this scenario has approximately 10 minutes and 30 seconds (630

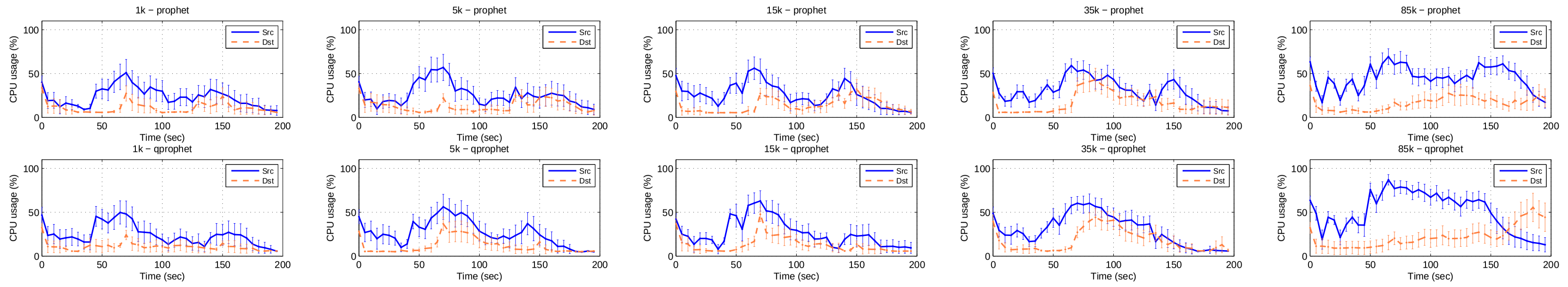


Figure 5.17: Scenario 2: CPU usage

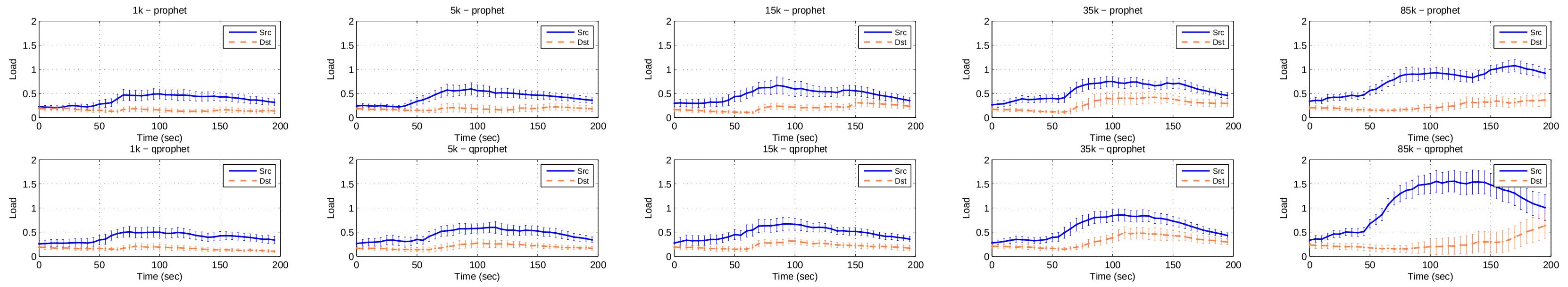
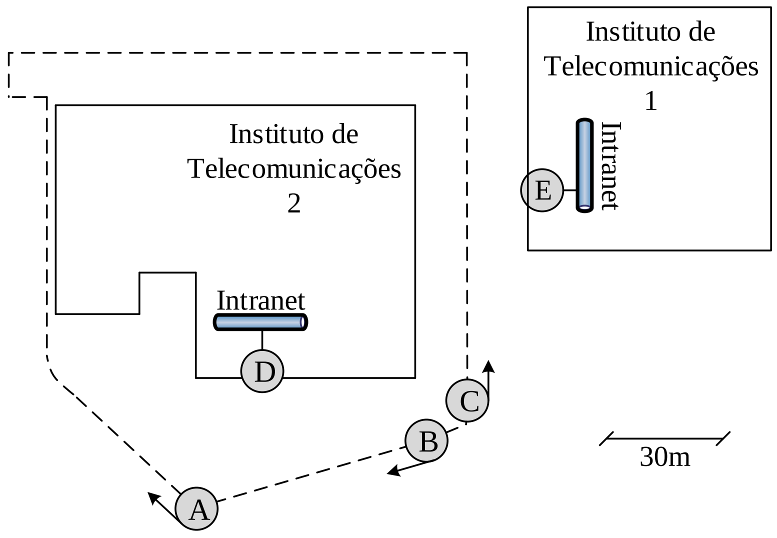


Figure 5.18: Scenario 2: Load average


Instituto de Telecomunicações 1
Intranet
E
Instituto de Telecomunicações 2
Intranet
D
C
B
A
30m
Figure 5.19: Scenario 3: Population Support - Boards Dynamics

seconds), i.e. the nodes A, B and C give three laps around building 2. The source node sends 240 files during the first 8 minutes, i.e. with an interval of 2 seconds between files. The experiments are performed for the following files size: 1KB, 5KB, 15KB and 35KB.

The synchronization of node D is performed using NTP because the board is connected to the Internet. The synchronization for the other nodes is performed using PTPd in all nodes, with node D as a master because it is considered a time reference, and other nodes are slaves. The experiments are launched and managed manually, due to the fact that not all boards are connected with a reliable connection to the master laptop.

5.6.4.3 Obtained Results

The results of the delivery ratio are presented in Figure 5.20. They show that for 1KB files, the delivery ratio is approximately 100% for both protocols. Note that for 5KB both delivery ratios are lower, but the ratio for Q-PRoPHET is better than PRoPHET. For 15KB the ratios are approximately equal with a better result for PRoPHET, and for 35KB the values of delivery ratio are lower for both protocols, but more pronounced for the Q-PRoPHET protocol.

The e2e delay results are presented in Figure 5.21. They show that, for files with 1KB, 5KB and 15KB the e2e, delays for Q-PRoPHET are lower than for PRoPHET, and they have a tendency to increase according to the file size. However, for 35KB, the e2e delay for Q-PRoPHET was worse than for PRoPHET.

These results show that is still room for improvements in Q-PRoPHET. In this specific case, the Q-PRoPHET has a lower number of transmissions than PRoPHET (Figure 5.22), which means that Q-PRoPHET always prioritizes one link until the end of the experiment. In this case, the link gets saturated faster due to the amount of files exchanged (35KB), which is due to the nodes mobility and the outdoor interferences, as can be seen in 5.20.

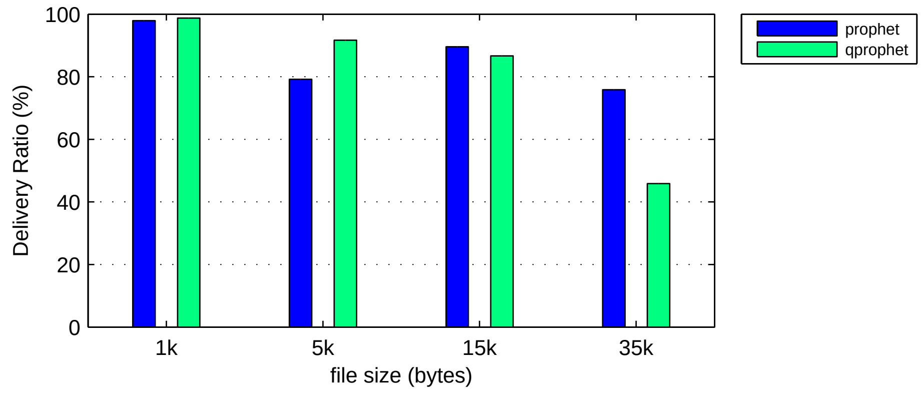


Figure 5.20: Scenario 3: Delivery Ratio

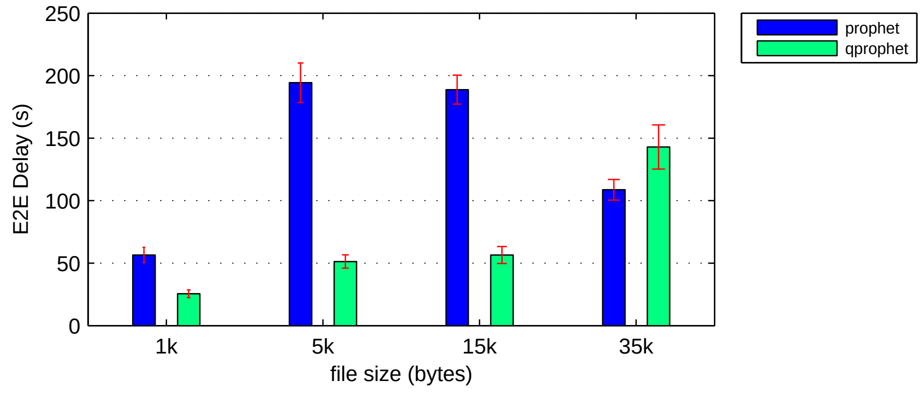


Figure 5.21: Scenario 3: e2e delay

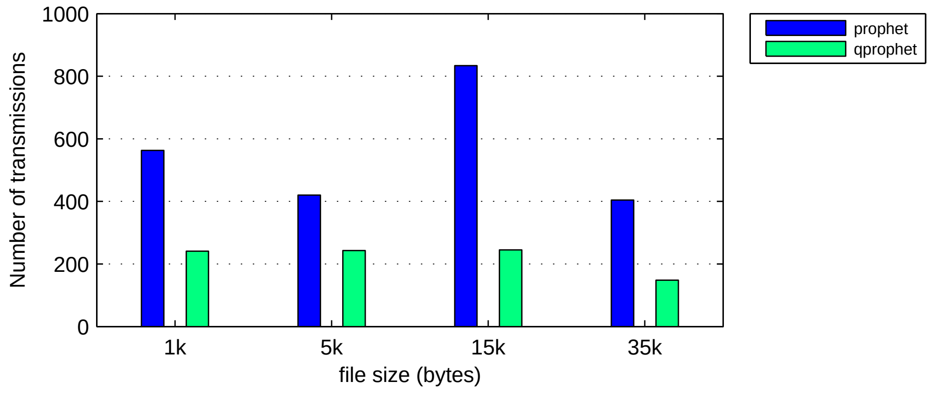


Figure 5.22: Scenario 3: Total transmissions per Experiment

The measurements of CPU usage and load average (1 minute) are presented in Figures 5.23 and 5.24. For 1KB and 5KB, the CPU and load values were always under 1 or near.

For 15KB and 35KB files, the CPU and load results are worse. Sometimes the load reaches value 2 which means that the Cambrias have difficulties performing their task. However, since the load does not diverge, the results are considered valid.

5.6.4.4 Quality Aging

Due to the fact that the results for the 35KB files for scenario 3 were not satisfactory, an improvement attempt was made to Q-PRoPHET. The improvement consists in the modification of the constant γ from 0.999 (default value) to 0.6 to decrease faster the quality aging in relation of the surrounding nodes. The idea is to force the quality measurement to recalculate the links quality of the surrounding nodes, in order to find new better quality alternatives to forward the bundles to destination nodes, rather than always prioritize few good quality links.

After running the experiment with changing γ value in Q-PRoPHET protocol, the obtained results are presented associated by the label '`35k (2)`'. The label '`35k (1)`' represents the previous result and it is presented again for comparison purposes.

The Q-PRoPHET delivery ratio in Figure 5.25 is now significantly higher than the previous version of Q-PRoPHET, which means that the variance of the constant γ affects strongly the obtained results.

The e2e delay in Figure 5.26 for Q-PRoPHET is now significantly lower (difference of approximately 60 seconds) than PRoPHET protocol. The improvements in the results of the delivery ratio and e2e delay are due to the fact that now the nodes consider themselves nodes with a lower quality in a very early stage. Moreover, Q-PRoPHET transmits more bundles than PRoPHET as it is shown in Figure 5.27, which contributes to better results.

Considering the tests that have been performed with different γ values in different scenarios, the best case is to adjust the constant γ value automatically according to the amount of surrounding neighbors. This means that, for dense scenarios, the aging of links should decrease very early, forcing the quality measurement module to update more frequently the surrounding links quality, and increasing the options to forward bundles over more quality links until reaching the destination. In sparse scenarios, higher values of γ introduce better results.

Figures 5.28 and 5.29 show the CPU consumption and load average (1 min.) with a confidence interval of 95%. By comparing the Q-PRoPHET results, it is possible to verify that the results of CPU and load are a bit worse for the case when γ is 0.6, due to the fact that the number of transmissions increase and consequently the resources are more overloaded.

5.7 Chapter Considerations

This chapter presented the testbeds used to evaluate the proposed approach, using Cambrias with openWRT OS. It presented also the integration and configuration of the software in the Cambrias.

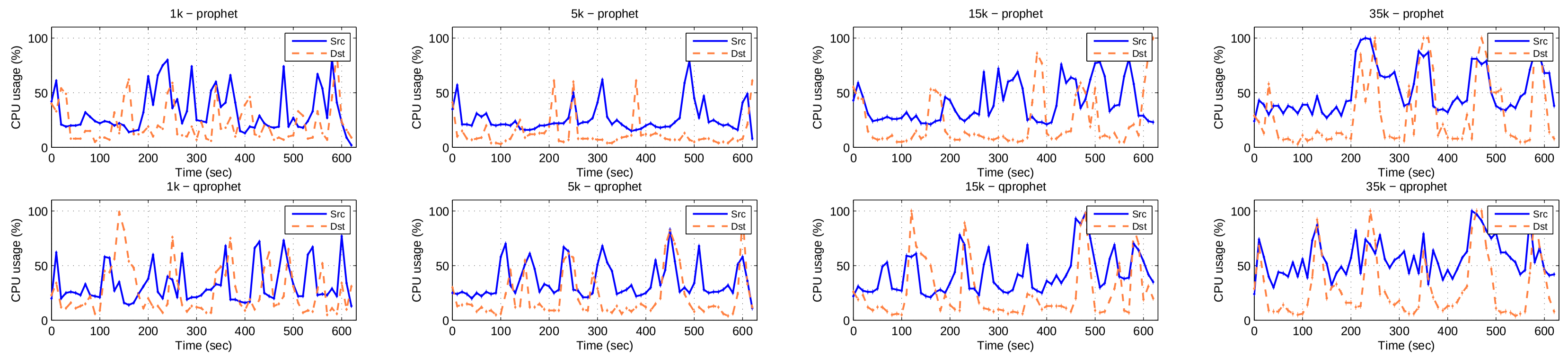


Figure 5.23: Scenario 3: CPU usage

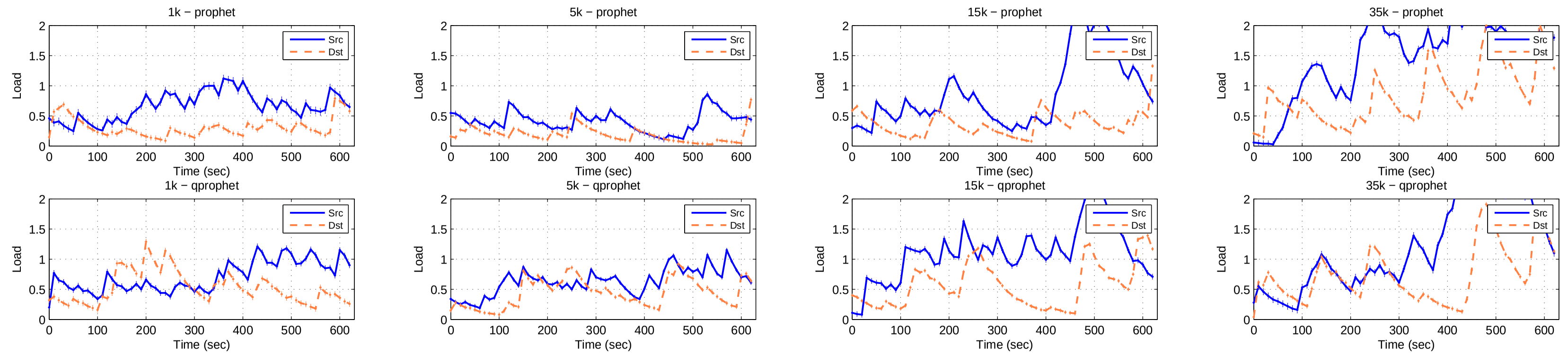


Figure 5.24: Scenario 3: Load average

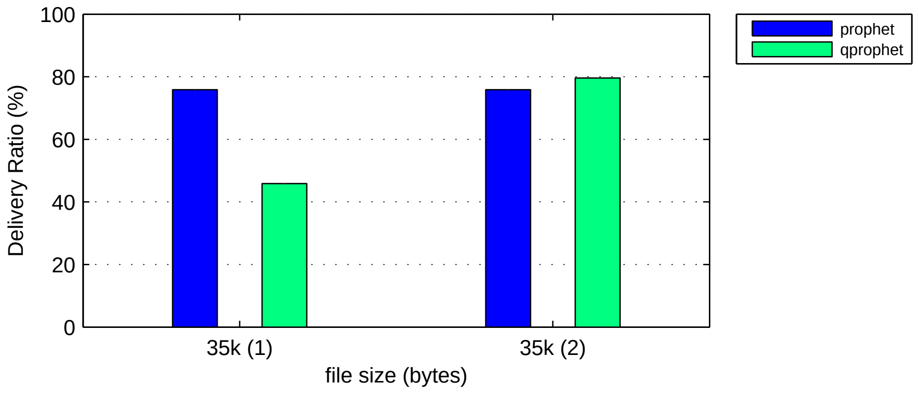


Figure 5.25: Scenario 3: Delivery Ratio

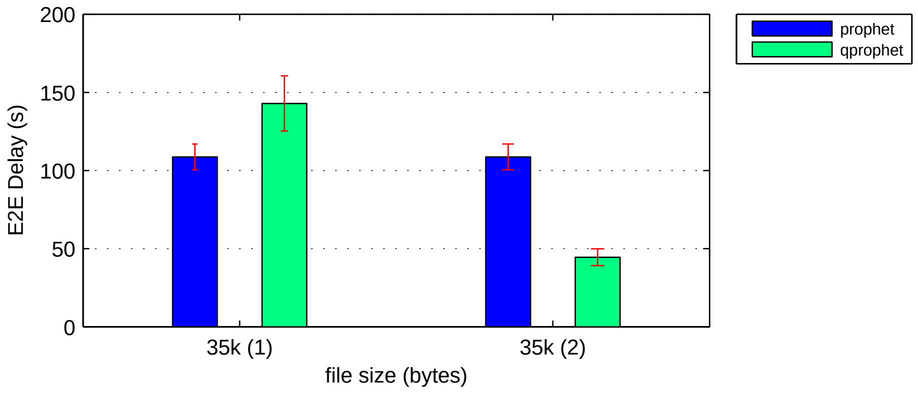


Figure 5.26: Scenario 3: e2e delay

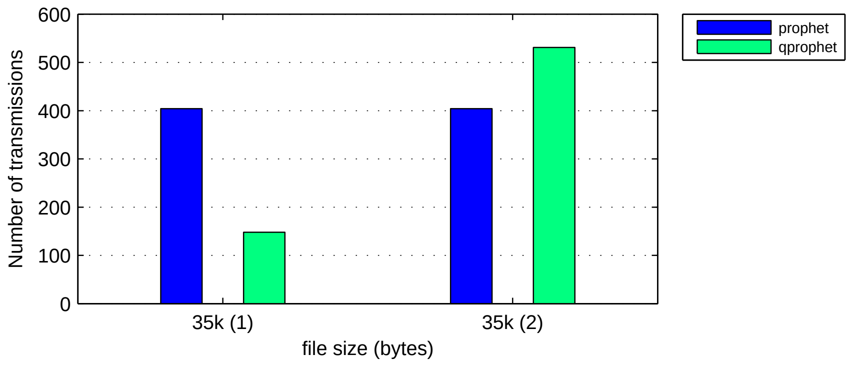


Figure 5.27: Scenario 3: Total transmissions per Experiment

Then, it presented the evaluation challenges. Some of the challenges were related to the fact that it is difficult to emulate the navy scenarios with real intermittence of the connections. Other challenges were due to the Cambrias' capabilities, which are not

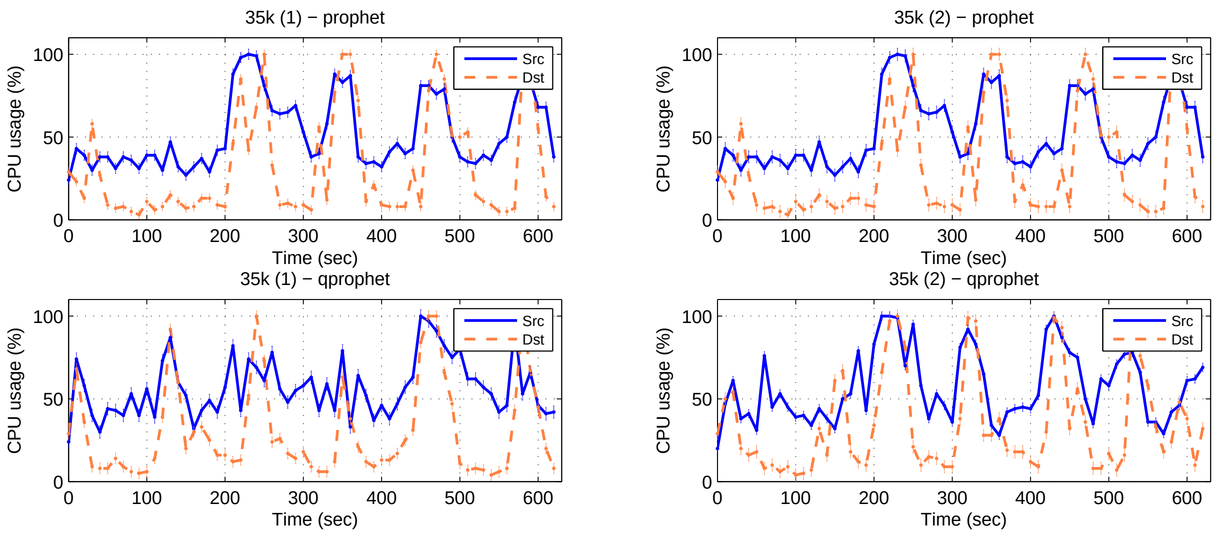


Figure 5.28: Scenario 3: CPU usage

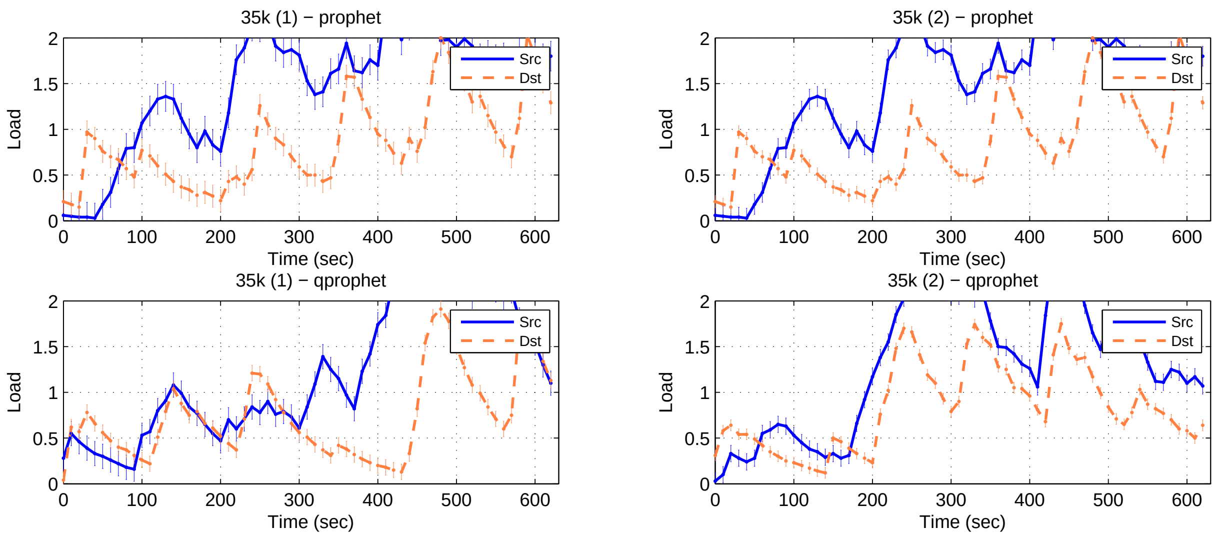


Figure 5.29: Scenario 3: Load average

sufficient to run IBR-DTN and the QCR efficiently, which forced to change some evaluation procedures to not overload the boards.

Therefore, the chapter presented the evaluation of the QCR API in an outdoor environment, which shows that the quality decreases with the distance as expected, but it also shows that the SSI has more sensitivity for lower distances and the LStab has more sensitivity to higher distances.

Finally, the evaluation of the routing approach in the several scenarios has been performed and analysed. The first two scenarios were indoor, and the last one was outdoor using real mobility of the nodes.

The obtained results show that a quality-based routing protocol should be used in DTNs: Q-PRoPHET has better performance than PRoPHET, a well-known routing protocol in DTNs, in terms of delivery ratio, e2e delay and packets transmission, which are critical parameters for the communication in navy operations.

However, in scenario 3 the obtained results of Q-PRoPHET with 35KB file size are worse than the same results of PRoPHET. To improve the Q-PRoPHET behavior, the value of γ was changed to 0.6 and the experiments were run again. The new results show that γ can affect the results: γ can be a good aging parameter to force the nodes to send more ou less bundles. Thus, the γ parameter should be automatically adjusted to work in sparse and dense scenarios. These results show that a quality-based routing should be applied in DTNs to improve their communications.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

During this work, it was referred that the participants of navy scenarios (mariners, boats, etc.) need to communicate between them to reach the common objectives. However, the environments of the navy operations are usually characterized by network fragmentation and intermittent connections, which affect the communication systems. In order to decrease the costs of the proprietary communication solutions that navy use, and to enable the possibility to insert easily new functionalities, this dissertation proposed a delay-tolerant quality-based routing protocol where the objective is to route information taking into account the connection quality of the links to be applied in off-the-shelf devices.

To develop the quality-based routing protocol, it was necessary to evaluate the quality of the links. The quality of the links is defined with a quality factor, which is a function of two metrics: the SSI and the LStab. The obtained results show that these metrics allow a good sensitivity to the quality factor in all distances, since the SSI has good sensitivity in lower distances, and LStab has good sensitivity in higher distances. The quality measurements are performed with the QCR, an API developed in this work, to measure the quality of the surrounding neighbors. To calculate the quality factor, the QCR gathers the SSI, contained in the radiotap headers, and it implements a beacons system to calculate the LStab. To provide the quality factor, the QCR has an IPC socket to communicate with other programs.

The quality-based routing protocol Q-PRoPHET uses the quality factor measured by the QCR. This routing protocol was based on PRoPHET, a well-known protocol developed for DTNs. The Q-PRoPHET was implemented into IBR-DTN and it was tested in a set of scenarios that emulate navy scenarios with the presence of intermittent connections. The same tests were also performed with PRoPHET in order to have a good reference to evaluate the obtained results.

The obtained results show that Q-PRoPHET has better performance than PRoPHET in terms of delivery ratio, e2e delay and packets transmission, which are critical parameters for the communication in navy operations. This shows that, for this type of scenarios, it is recommended to use routing algorithms based on the quality of the wireless links to perform routing decisions.

It was also concluded that it is a big challenge to emulate real conditions of intermittence, or maintain these conditions to perform several tests in the same conditions to obtain results with a small confidence interval, due to the fact that these scenarios are unstable, i.e. they vary their characteristics along the time. To emulate the scenarios, different techniques are adopted separately: use an intermittence script in the boards to vary their antennas' transmission power, perform the tests without antennas to control better the links failures, or use real mobility. Using these approaches, we were able to run the tests and obtain results with good confidence values.

Another challenge was related to the boards capabilities to run the programs and to perform the tests. First, the initial version from QCR consumed all the CPU resources by analyzing all IEEE802.11 beacon frames in the media. To solve this issue, it was implemented a beacons system in the QCR, and the tests were performed in a way that the load has been distributed over time.

Relatively to IBR-DTN, it is concluded that it is not adequate to perform this type of routing decisions based on the MAC/PHY layer. For example, this routing protocol deals directly with the quality of the connections, and if the connection does not exist, the information about it should be integrated directly with the Discovery Agent, i.e. in this case the Discovery Agent should be implemented or depend directly from the quality values and not the opposite. In this work, the IBR-DTN source code was changed to maintain the IBR-DTN neighbors tables concordant with the QCR tables, i.e. when IBR-DTN detects a new neighbor, it adds the neighbor to QCR; and when IBR-DTN looses a neighbor, it removes it from QCR. This allows to solve the problem, but there is duplicated information and it is necessary more processing.

Another characteristic of IBR-DTN is that it is not adequate for this type of dynamic routing, because a node that contains several bundles just verifies if they should be sent when they are generated, when a new contact is made or when it receives a new routing table from another node. However, in Q-PRoPHET the local tables may change more frequently and the changes are not often associated to the events that verify if a node should send a certain bundle. Due to this, the IBR-DTN is not adequate to implement routing protocols that operate directly with physical conditions due to their dynamics, because the quality values change very fast and the node cannot verify always its storage to decide if it should send the bundles. When the quality of the link changes, the routing changes, but the storage is not always checked. However, due to the process of tables exchange, the bundles are frequently checked.

Finally, this work allows to conclude that the quality-based routing protocols should be an investment to apply in DTNs to work in scenarios with navy constrains. The Q-PRoPHET proved that links quality-based routing can indeed improve the results in networks with this type of constrains. However, there is still work to be done in this promising investigation area.

6.2 Future Work

For future work the Q-PRoPHET should be tested in other boards with more CPU and memory resources, and with more demanding scenarios, in order to verify if the results for high traffic are good and if the protocol may be improved.

It is also suggested to implement a mechanism that prevents the case of link saturation. For example, if a node selects a preferred link to forward information, based on its quality, it may get overloaded because it is considered the best link.

Another important aspect is the DTN implementation. In this case, it is recommended to implement a specific DTN implementation specially for this type of scenarios where the events to send bundles are different, and where the discovery methods should be also different, mainly the Discovery Agent and the base router.

Another suggestion for future work is to perform the tests with a noise generator in order to create real conditions of intermittence and easily replicate them, or to perform the tests with a rail in order to create equal conditions of mobility in different tests and to run them automatically without human intervention to impose mobility.

There are also several suggestions to improve Q-PRoPHET. One of them is to perform a further study about the influence of γ in different scenarios. γ has the capability to decrease faster the quality in the nodes' table. Therefore, future work will consider the proposal of an approach to adjust automatically γ value according to the nodes density.

It is also relevant to consider other metrics to evaluate the quality, for example the throughput and the saturation level of the links.

Another approach is the proposal of a method to evaluate if there is an e2e path between the sender and destination nodes. With this method, Q-PRoPHET would be capable to perform quality-based routing to send the bundle when there is an e2e path, like an ad-hoc behavior where the objective is to benefit from the best quality path. But, if there is not an e2e path to the destination, it should be applied a different routing protocol to move the bundle to better delivers, i.e. move it to the nodes that (probably) will encounter the destination, or to even trigger other technologies to send the information if it is urgent.

Bibliography

[1] C. Hunt, *TCP/IP network administration.* " O'Reilly Media, Inc.", 2002, vol. 2.

[2] S. Farrell, V. Cahill, D. Geraghty, I. Humphreys, and P. McDonald, "When tcp breaks: Delay-and disruption-tolerant networking," *Internet Computing, IEEE*, vol. 10, no. 4, pp. 72–78, 2006.

[3] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-tolerant network architecture: the evolving interplanetary internet," Working Draft, IETF Secretariat, Internet-Draft draft-irtf-ipnrg-arch-01, August 2002. [Online]. Available: https://tools.ietf.org/html/draft-irtf-ipnrg-arch-01

[4] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, E. Travis, and H. Weiss, "Interplanetary internet (ipn): architectural definition," 2001.

[5] A. McMahon and S. Farrell, "Delay-and disruption-tolerant networking," *IEEE Internet Computing*, no. 6, pp. 82–87, 2009.

[6] K. Fall and S. Farrell, "Dtn: an architectural retrospective," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 5, pp. 828–836, 2008.

[7] K. Fall, "Disruption tolerant networking for heterogeneous ad-hoc networks," in *Military Communications Conference, 2005. MILCOM 2005. IEEE.* IEEE, 2005, pp. 2195–2201.

[8] A. S. Pentland, R. Fletcher, and A. Hasson, "Daknet: Rethinking connectivity in developing nations," *Computer*, vol. 37, no. 1, pp. 78–83, 2004.

[9] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications.* ACM, 2003, pp. 27–34.

[10] R. Monteiro, L. Guedes, T. Condeixa, F. Neves, S. Sargento, L. Guardalben, and P. Steenkiste, "Lessons learned from a real vehicular network deployment of delay-tolerant networking," in *Communication Workshop (ICCW), 2015 IEEE International Conference on Communications.* IEEE, 2015, pp. 2489–2494.

[11] Z. Lu and J. Fan, "Delay/disruption tolerant network and its application in military communications," in *Computer design and applications (ICCDA), 2010 international conference on*, vol. 5. IEEE, 2010, pp. V5–231.

[12] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *Wireless networks*, vol. 8, no. 5, pp. 481–494, 2002.

[13] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-Tolerant Networking Architecture," RFC 4838 (Informational), Internet Engineering Task Force, Apr. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4838.txt

[14] A. V. Vasilakos, Y. Zhang, and T. Spyropoulos, *Delay tolerant networks: Protocols and applications.* CRC press, 2011.

[15] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," RFC 3986 (INTERNET STANDARD), Internet Engineering Task Force, Jan. 2005, updated by RFCs 6874, 7320. [Online]. Available: http://www.ietf.org/rfc/rfc3986.txt

[16] K. Scott and S. Burleigh, "Bundle Protocol Specification," RFC 5050 (Experimental), Internet Engineering Task Force, Nov. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc5050.txt

[17] M. H. Jain and R. Patra, "Implementing delay tolerant networking," *Intel Research, Berkeley, Technical Report, IRB-TR-04-020*, Dec. 2004.

[18] S. Symington, S. Farrell, H. Weiss, and P. Lovell, "Bundle Security Protocol Specification," RFC 6257 (Experimental), Internet Engineering Task Force, May 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6257.txt

[19] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," RFC 5905 (Proposed Standard), Internet Engineering Task Force, Jun. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5905.txt

[20] PTPd. (2015, Sep.) Precision time protocol daemon. [Online]. Available: http://ptpd.sourceforge.net/

[21] K. Fall, "A delay-tolerant network architecture for challenged internets," *Intel Research Technical, Report IRB-TR-03-003*, Feb. 2003.

[22] D. Steedman, *Abstract syntax notation one (ASN. 1): the tutorial and reference.* Technology appraisals, 1993.

[23] ASN, ITUT, "Encoding rules: Specification of basic encoding rules (ber), canonical encoding rules (cer) and distinguished encoding rules (der)," Technical report, International Telecommunication Union, Tech. Rep., 2002.

[24] E. Arias and B. Guinot, "Coordinated universal time utc: historical background and perspectives," in *Journees systemes de reference spatio-temporels*, 2004.

[25] Delay-Tolerant Networking Research Group (DTNRG). (2015, Jan.) Dtn2. [Online]. Available: https://sites.google.com/site/dtnresgroup/home/code/dtn2documentation

[26] E. Davis and A. Doria, "D2.2: Functional specification for dtn infrastructure software," Folly Consulting and LTU, Tech. Rep., 2010.

[27] M. Demmer and K. Fall, "Dtlsr: delay tolerant routing for developing regions," in *Proceedings of the 2007 workshop on Networked systems for developing regions.* ACM, 2007, p. 5.

[28] A. Seth, P. Darragh, S. Liang, Y. Lin, and S. Keshav, "An architecture for tetherless communication," *Disruption Tolerant Networking*, vol. 5142, 2005.

[29] Institut für Betriebssysteme und Rechnerverbund. (2015, Jan.) Ibr-dtn. [Online]. Available: https://trac.ibr.cs.tu-bs.de/project-cm-2012-ibrdtn

[30] M. Doering, S. Lahde, J. Morgenroth, and L. Wolf, "Ibr-dtn: an efficient implementation for embedded systems," in *Proceedings of the third ACM workshop on Challenged networks.* ACM, 2008, pp. 117–120.

[31] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf, "Ibrdtn: A lightweight, modular and highly portable bundle protocol implementation," in *Electronic Communications of the EASST.* Citeseer, 2011.

[32] D. Ellard, R. Altmann, A. Gladd, and D. Brown, "Dtn ip neighbor discovery (ipnd)," Working Draft, IETF Secretariat, Internet-Draft draft-irtf-dtnrg-ipnd-02, November 2012. [Online]. Available: https://tools.ietf.org/html/draft-irtf-dtnrg-ipnd-02

[33] M. Demmer, J. Ott, and S. Perreault, "Delay-Tolerant Networking TCP Convergence-Layer Protocol," RFC 7242 (Experimental), Internet Engineering Task Force, Jun. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7242.txt

[34] H. Kruse, S. Jero, and S. Ostermann, "Datagram Convergence Layers for the Delay- and Disruption-Tolerant Networking (DTN) Bundle Protocol and Licklider Transmission Protocol (LTP)," RFC 7122 (Experimental), Internet Engineering Task Force, Mar. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7122.txt

[35] "Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirement part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans)," *IEEE Std 802.15.4a-2007 (Amendment to IEEE Std 802.15.4-2006)*, pp. 1–203, 2007.

[36] Veniam®. (2015, Sep.) Veniam, an internet of moving things. [Online]. Available: https://veniam.com/

[37] A. Vahdat, D. Becker *et al.*, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Duke University, Tech. Rep., 2000.

[38] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," in *Service Assurance with Partial and Intermittent Resources.* Springer, 2004, pp. 239–254.

[39] A. Lindgren, A. Doria, E. Davies, and S. Grasic, "Probabilistic Routing Protocol for Intermittently Connected Networks," RFC 6693 (Experimental), Internet Engineering Task Force, Aug. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6693.txt

[40] M. Georgescu, T. Sahara, M. Ashar, H. Izumikawa, Y. Onogi, M. Tamai, and S. Kashihara, "Performance analysis of file transmission in dtn2 and ibr-dtn," 2013.

[41] R. Beuran, S. Miwa, and Y. Shinoda, "Performance evaluation of dtn implementations on a large-scale network emulation testbed," in *Proceedings of the seventh ACM international workshop on Challenged networks.* ACM, 2012, pp. 39–42.

[42] W.-B. Pöttner, J. Morgenroth, S. Schildt, and L. Wolf, "Performance comparison of dtn bundle protocol implementations," in *Proceedings of the 6th ACM workshop on Challenged networks.* ACM, 2011, pp. 61–64.

[43] C. Gavoille, "Routing in distributed networks: Overview and open problems," *ACM SIGACT News*, vol. 32, no. 1, pp. 36–52, 2001.

[44] S. Jain, K. Fall, and R. Patra, *Routing in a delay tolerant network.* ACM, 2004, vol. 34, no. 4.

[45] E. P. Jones and P. A. Ward, "Routing strategies for delay-tolerant networks," *Submitted to ACM Computer Communication Review (CCR)*, 2006.

[46] W. Moreira and P. Mendes, "Survey on opportunistic routing for delay/disruption tolerant networks," 2010.

[47] Z. Zhang, "Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges," *IEEE Communications Surveys & Tutorials*, vol. 1, no. 8, pp. 24–37, 2006.

[48] A. Balasubramanian, B. Levine, and A. Venkataramani, "Dtn routing as a resource allocation problem," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 373–384, 2007.

[49] L. Song and D. F. Kotz, "Evaluating opportunistic routing protocols with large realistic contact traces," in *Proceedings of the second ACM workshop on Challenged networks.* ACM, 2007, pp. 35–42.

[50] S. C. Nelson, M. Bakht, and R. Kravets, "Encounter-based routing in dtns," in *INFOCOM 2009, IEEE.* IEEE, 2009, pp. 846–854.

[51] R. D'souza and J. Jose, "Routing approaches in delay tolerant networks: A survey," *International Journal of Computer Applications*, vol. 1, no. 17, pp. 8–14, 2010.

[52] T. Spyropoulos, R. N. Rais, T. Turletti, K. Obraczka, and A. Vasilakos, "Routing for disruption tolerant networks: taxonomy and design," *Wireless networks*, vol. 16, no. 8, pp. 2349–2370, 2010.

[53] A. Mtibaa, M. May, C. Diot, and M. Ammar, "Peoplerank: Social opportunistic forwarding," in *INFOCOM, 2010 Proceedings IEEE.* IEEE, 2010, pp. 1–5.

[54] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks," *Mobile Computing, IEEE Transactions on*, vol. 10, no. 11, pp. 1576–1589, 2011.

[55] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing.* ACM, 2007, pp. 32–40.

[56] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing.* ACM, 1987, pp. 1–12.

[57] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *ACM SIGMOBILE mobile computing and communications review*, vol. 7, no. 3, pp. 19–20, 2003.

[58] A. Lindgren and K. S. Phanse, "Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks," in *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on.* IEEE, 2006, pp. 1–10.

[59] S. Grasic, E. Davies, A. Lindgren, and A. Doria, "The evolution of a dtn routing protocol-prophetv2," in *Proceedings of the 6th ACM workshop on Challenged networks.* ACM, 2011, pp. 27–30.

[60] T.-K. Huang, C.-K. Lee, and L.-J. Chen, "Prophet+: An adaptive prophet-based routing protocol for opportunistic network," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on.* IEEE, 2010, pp. 112–119.

[61] S. D. Han and Y. W. Chung, "An improved prophet routing protocol in delay tolerant network," *The Scientific World Journal*, vol. 2015, 2015.

[62] P. Sok and K. Kim, "Distance-based prophet routing protocol in disruption tolerant network," in *ICT Convergence (ICTC), 2013 International Conference on.* IEEE, 2013, pp. 159–164.

[63] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking.* ACM, 2005, pp. 252–259.

[64] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks." in *INFOCOM*, vol. 6, 2006, pp. 1–11.

[65] J. Leland and I. Porche, "Future army bandwidth needs and capabilities," *Rand Corporation*, 2004.

[66] J. L. Burbank, P. F. Chimento, B. K. Haberman, and W. T. Kasch, "Key challenges of military tactical networking and the elusive promise of manet technology," *Communications Magazine, IEEE*, vol. 44, no. 11, pp. 39–45, 2006.

[67] European Telecommunications Standards Institute. (2015, Mar.) Tetra. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/tetra

[68] J. Nielson. (2015, Mar.) Terrestrial trunked radio. [Online]. Available: http://www.uwplatt.edu/files/csse/courses/prev/csse411-materials/s12/TETRApaper_Joe_Nielson.docx

[69] P. Stavroulakis, *Terrestrial trunked radio-TETRA: a global security tool.* Springer Science & Business Media, 2007.

[70] K.-W. Chin, "The behavior of manet routing protocols in realistic environments," in *Communications, 2005 Asia-Pacific Conference on.* IEEE, 2005, pp. 906–910.

[71] J.-M. Choi and Y.-B. Ko, "A performance evaluation for ad hoc routing protocols in realistic military scenarios," in *Proceedings of the 9th International Conference on Cellular and Intelligent Communications (CIC 2004)*, 2004.

[72] S. Parikh and R. C. Durst, "Disruption tolerant networking for marine corps condor," in *Military Communications Conference, 2005. MILCOM 2005. IEEE.* IEEE, 2005, pp. 325–330.

[73] T. Jonson, J. Pezeshki, V. Chao, K. Smith, and J. Fazio, "Application of delay tolerant networking (dtn) in airborne networks," in *Military Communications Conference, 2008. MILCOM 2008. IEEE.* IEEE, 2008, pp. 1–7.

[74] C. Rigano, K. Scott, J. Bush, R. Edell, S. Parikh, R. Wade, and B. Adamson, "Mitigating naval network instabilities with disruption toler," in *Military Communications Conference, 2008. MILCOM 2008. IEEE.* IEEE, 2008, pp. 1–7.

[75] IEEE Standards Association and others, "802.11-2012-ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," Mar 2012.

[76] Microsoft®. (2015, Mar.) How 802.11 wireless works. [Online]. Available: https://technet.microsoft.com/en-us/library/cc757419(v=ws.10).aspx

[77] L. Guardalben, "Communication between nodes for autonomic and distributed management," Ph.D. dissertation, Universidade de Aveiro, 2014.

[78] Linux Wireless wiki. (2015, Mar.) Documentation for the linux wireless (ieee-802.11) subsystem. [Online]. Available: https://wireless.wiki.kernel.org/

[79] R. Kuschnig, E. Yanmaz, I. Kofler, B. Rinner, and H. Hellwagner, *Profiling IEEE 802.11 Performance on Linux-based Networked Aerial Robots.*

[80] Linux Wireless wiki. (2015, Mar.) Documentation - cfg80211. [Online]. Available: https://wireless.wiki.kernel.org/en/developers/documentation/cfg80211

[81] ——. (2015, Mar.) Documentation - mac80211. [Online]. Available: https://wireless.wiki.kernel.org/en/developers/documentation/mac80211

[82] ——. (2015, Mar.) Documentation - nl80211. [Online]. Available: https://wireless.wiki.kernel.org/en/developers/documentation/nl80211

[83] G. P. Zanetti and C. E. Palazzi, "Non-invasive node detection in ieee 802.11 wireless networks," in *Wireless Days (WD), 2010 IFIP.* IEEE, 2010, pp. 1–5.

[84] Linux Wireless wiki. (2015, Apr.) Existing linux wireless drivers. [Online]. Available: https://wireless.wiki.kernel.org/en/users/drivers

[85] J. Berg. (2015, Feb.) Radiotap - radiotap.org. [Online]. Available: http://www.radiotap.org/

[86] Linux Wireless wiki. (2015, May.) About iw. [Online]. Available: https://wireless.wiki.kernel.org/en/users/documentation/iw

[87] Debian Config. (2015, Ago.) iwconfig. [Online]. Available: https://wiki.debian.org/iwconfig

[88] The Tcpdump Group. (2015, Feb.) Tcpdump and libpcap. [Online]. Available: http://www.tcpdump.org/

[89] Wireshark Foundation. (2015, Oct.) Wireshark. [Online]. Available: https://www.wireshark.org/

[90] The Tcpdump Group. (2015, Feb.) Manpage of pcap. [Online]. Available: http://www.tcpdump.org/manpages/pcap.3pcap.html

[91] S. McCanne and V. Jacobson, "The bsd packet filter: A new architecture for user-level packet capture," in *Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings.* USENIX Association, 1993, pp. 259–270.

[92] L. Deri *et al.*, "Improving passive packet capture: Beyond device polling," in *Proceedings of SANE*, vol. 2004. Amsterdam, Netherlands, 2004, pp. 85–93.

[93] L. M. Garcia, "Programming with libpcap - sniffing the network from our own application," *Hakin9-Computer Security Magazine*, pp. 38–46, 2008.

[94] M. Beeler, R. Gosper, and R. Schroeppel, "Hakmem. memorandum 239," *Artificial Intelligence Laboratory, MIT, Feb*, 1972.

[95] A. Oram and G. Wilson, "Beautiful code: Leading programmers explain how they think (theory in practice)," 2007.

[96] Institut für Betriebssysteme und Rechnerverbund. (2015, Feb.) Ibr-dtn 1.0.0 doxygen. [Online]. Available: https://www.ibr.cs.tu-bs.de/projects/ibr-dtn/doxygen/1.0.0/

[97] Gateworks®. (2015, Jan.) Cambria gw2358-4 single board computer. [Online]. Available: http://www.gateworks.com/product/item/cambria-gw2358-4-network-processor

[98] *Cambria Network Computer - Operating Manual For GW2358-4 Network Processor*, GATEWORKS®, 08 2008, rev. 01.

[99] Atheros Communications®. (2015, Jan.) Ar5004g - 802.11b/g wlan solution. [Online]. Available: https://wikidevi.com/files/Atheros/specsheets/AR5004G.pdf

[100] OpenWrt. (2015, Feb.) Openwrt wireless freedom. [Online]. Available: https://openwrt.org/

[101] How-To Geek, LLC. (2015, Jun.) Understanding the load average on linux and other unix-like systems. [Online]. Available: http://www.howtogeek.com/194642/understanding-the-load-average-on-linux-and-other-unix-like-systems/

[102] NTP. (2015, Sep.) Network time protocol. [Online]. Available: http://ptpd.sourceforge.net/