Shaping Dataflows with Ever-changing Datarates in Tactical Networks

Master Thesis

Pooja Hanavadi Balaraju

This work was submitted to the

Chair of Communication and Distributed Systems

RWTH Aachen University, Germany

Adviser(s):

Dr. Roberto Rigolin F Lopes
Dr. Paulo Henrique Rettore Lopes
Martin Serror, M. Sc.

Examiners:

Prof. Dr.-Ing. Klaus Wehrle
Prof. Dr.-Ing. Frank Kurth

Registration date: 2019-10-29
Submission date: 2020-06-04

Zentrales Prüfungsamt/Central Examination Office

Eidesstattliche Versicherung
Statutory Declaration in Lieu of an Oath

Hanavadi Balaraju,Pooja | 383 256

Name, Vorname/Last Name, First Name | Matrikelnummer (freiwillige Angabe) Matriculation No. (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende ~~Arbeit~~/~~Bachelorarbeit~~/ Masterarbeit* mit dem Titel

I hereby declare in lieu of an oath that I have completed the present ~~paper~~/~~Bachelor thesis~~/Master thesis* entitled

Shaping Dataflows with Ever-changing Datarates in Tactical Networks

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

independently and without illegitimate assistance from third parties (such as academic ghostwriters). I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Aachen, 04.06.2020

Ort, Datum/City, Date | Unterschrift/Signature

*Nichtzutreffendes bitte streichen

*Please delete as appropriate

Belehrung:
Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

Para. 156 StGB (German Criminal Code): False Statutory Declarations
Whoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence
(1) If a person commits one of the offences listed in sections 154 through 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.
(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Aachen, 04.06.2020

Ort, Datum/City, Date | Unterschrift/Signature

Abstract

Tactical Network (TN) supports military communications by facilitating information exchange to accomplish mission-oriented tasks. The users are the military personnel who communicate via verbal, written, audio, or video messages. The magnitude of user behavior depends on events/emergencies like warfare, the conduct of combat, evacuation, and many more whose occurrence is highly unpredictable. The network mostly comprises of Mobile Adhoc Network (MANET) which has frequent node mobility and topology changes. Moreover, the underlying networks are characterized as *ever-changing* and unpredictable because of low bandwidth, high delay, and frequent disruptions. The tactical system such as middleware, proxy, or web service facilitates the data transmission between users through the network. However, the unpredictable nature of the network poses a challenge to the data transmission process. Therefore, there is a need for a system to be robust to network changes by reducing the impact on data transmission with a suitable adaptation mechanism. This study focuses on *improving the robustness of the tactical system to the network variation and disconnection*. Recent studies have implemented adaptation mechanisms in middlewares and proxies for accomplishing certain tasks like resource utilization, modification of transmission rates, and many others. This study concentrates on shaping the data flow to avoid congestion in case of network variation and identification of link disconnection to improve network awareness. Since, testing the system in real military scenarios is expensive, time-constrained, and has limitations for replicating the test cases for further analysis, this research is carried out in a laboratory environment. This thesis was built on the hypothesis that a suitable store-and-forward mechanism such as a hierarchical queuing model consisting of a message, packet, and radio buffer can be used to implement and test the robustness of the system. A control mechanism called Internet-Packet-Interval (IPI) was implemented to introduce a delay between the packet transmission based on the radio and link parameters obtained from the cross-layer exchange. Experimental evaluations were carried out by varying the data rates in the network and the results show that this mechanism is robust in dynamically adapting the data flow to changes in the network and system (radio). On the other hand, a machine learning approach and a time-based anomaly model are implemented to improve the network awareness of the system for link disconnection identification. Experimental scenarios were conducted by simulating link disconnection. The results of the machine learning approach did not provide sufficient arguments to explore this approach deeply. On the contrary, the time-based approach provided a suitable alternative whose predictions attained 89% precision that could be used to minimize the impact of link disconnection on the system. Thus, with experimental results, the robustness of the system to network changes and disconnection are highlighted with ever-changing network scenarios in the laboratory environment.

Acknowledgments

I would like to express my sincere gratitude to my supervisors Prof.Dr-Ing. Klaus Wehrle and Prof.Dr-Ing. Frank Kurth for their continuous support throughout my Master thesis research. Besides my supervisors, I would like to thank my thesis advisors: Dr. Roberto Rigolin Ferriera Lopes, Dr. Paulo Henrique Rettore Lopes, and Martin Serror for their patience, motivation, and guidance through the entire process of my Thesis.

My sincere thanks to the student researchers in Fraunhofer FKIE group: Adrian Toribio Silva, Sharath Maligera, and Johannes Lovenich for their support and insightful comments that enriched my learning experience. Last but not the least, I would like to thank my family for their continued patience and support always.

Contents

1

Introduction

Tactical Network (TN) supports military communications by facilitating information exchange to accomplish mission-oriented tasks [10, 17]. The users of tactical networks include military personnel, soldiers, convoys, and medical team who communicate via text, audio, or video messages. The magnitude of user behavior depends on events/emergencies like warfare, the conduct of combat, evacuation, and many more, whose occurrence is highly unpredictable. On the other hand, the backbone of the underlying network is Mobile Adhoc Network (MANET), which is distributed, configuring nodes that lack infrastructure [38]. They support dynamic topology changes and mobility of the nodes which makes them suitable for tactical networks. The task of the tactical system is to connect the users to the network and facilitate data transmission by keeping both the users and network opaque to each other. In other words, the users are unaware of the network conditions and vice-versa.

This study focuses on the interaction of the tactical system with the underlying network conditions. The network mostly consists of wireless links that have lower reliability with data delivery [46] and limited bandwidth [15]. This is reflected in the shortcomings of the TN such as network heterogeneity, low bandwidth, high latency, and node mobility [40, 48, 7, 52, 18]. Besides, they are subjected to frequent topology changes and disconnection, which characterizes the tactical networks to be volatile and *ever-changing* [55, 18, 30].

The ever-changing behavior of the system makes it difficult for the system to handle data transmission. For example, low bandwidth can result in data congestion and frequent disruption leads to packet loss. Therefore, many research studies are carried out in the interest of improving the robustness of the tactical system to the varying network conditions and disconnection. For instance, the development of middleware to adapt to the transmission rates [19, 10], transport protocol to choose suitable path [41], effective resource utilization [18], and shaping of the data flow [33]. In this research study, potential improvement in [33] is identified to increase the robustness of the system to varying network conditions and disconnection.

1.1 Research question

Therefore, this thesis aims to address the following research question.

- How to improve the robustness of the system to identify network variation/disconnection and subsequently adapt to the network?

The research began with the hypothesis that *the features of a store-and-forward mechanism such as the queuing model [33] can be leveraged to improve the robustness of the system to a) network changes by shaping the data flow based on network and system awareness and b) identify link disconnection.*

The next step was to test the tactical system rigorously against military network scenarios. However, testing with real military scenarios are expensive [13] and not easy to replicate [35] due to data confidentiality. In this sense, the testing process was narrowed down to using simulation/emulations in the laboratory environment. However, this process requires scenarios and data to replicate the tests in the laboratory [54, 51, 53]. Some of the recent studies try to solve this problem by acquiring real-time data and simulating the traffic into virtual platforms [13, 49], use Radio Frequency (RF) models [53, 54] and statistical distributions [45, 37, 32]. However, acquiring real-time data is still a challenging task and the RF models might not be applicable to all the radios as the configurations of the devices vary based on the manufacturers. Therefore, statistical distributions are used for the reasons of easy replication and no pre-requisite installation.

1.2 Goals

To improve the robustness of the system, the model of the hierarchy of queues namely the message queue, packet queue, and radio buffer which complement each other by sharing contextual information using cross-layer information exchange [33] were used. This study investigated on improving and testing the robustness of the system concerning two scenarios, namely, varying data rates in the network and link disconnections in the network. This is discussed in detail as follows.

- Varying data rates in the network: introduction of a delay between the packet transmission to regulate/shape the data flow based on parameters from multiple layers extracted using cross-layer information exchange. This helps for efficient usage of the system resources and reduce data congestion. In addition, an approach is proposed to compute the link data rate based on the cross-layer information exchange without adding any overhead in the network.
- Link disconnections in the network: Several features are extracted from the different layers of the queuing model which are used to classify if the link is connected or not. Keeping this as the motivation, the link is distinguished using a binary classifier of the machine learning approach and time-based anomaly detection in packet interception. The latter also predicts a list of packets lost during the disconnection. Both approaches have a common goal of improving the network awareness of the system for link disconnection identification.

1.3 Contributions

Suitable adaptation mechanisms are implemented to increase the robustness of the system to network variation and disconnection. The experimental results are used to identify the contributions of this research study as follows:

- To reduce the impact of the varying data rates in the network link data rate computation and *Inter-Packet-Interval* are implemented. The former calculates the link data rate without adding overhead in the network. This computation can be used to shape the data flow or given as an input to the link estimation of the routing protocol. The latter is experimentally proven to increase the robustness of the system by dynamically shaping the data flow to the network changes and reduce data congestion. This methodology can also be applied to other frameworks having an interface to the radio and cross-layer information exchange.
- The results of the machine learning model did not provide sufficient arguments to explore this approach deeply. However, this methodology can be used by other researchers with potential improvements to the model such as access to more input features and advanced data processing techniques. On the other hand, the time-based anomaly detection mechanism can be implemented in any framework with a suitable packet interceptor. The precision of predicting the lost packets attained 89% precision, using which the system can re-transmit lost packets to reduce the packet loss. This mechanism has an advantage of not adding any overhead in the network which makes it suitable for bandwidth-constrained networks.

1.4 Structure of the text

The structure of the thesis text is as follows. Chapter 2 discusses the fundamental concepts and taxonomy in the tactical domain. Chapter 3 outlines the problem statement that is aimed to solve in this study. Chapter 4 discusses some of the recent developments and solutions proposed for a similar problem. In addition, an overview of (TACTICS) project that implements the hierarchical queuing model of the message queue, packet queue, and radio buffer to enable the store-and-forward mechanism is explained. Chapter 5 translates the hypothesis into design considerations within the framework of the TACTICS project. Chapter 6 discusses the methodology of implementing the delay mechanism for shaping the data flow and two approaches to identify link disconnection. With quantitative results in Chapter 7, the hypothesis is confirmed to show that the system is robust to handle network variations and disconnection of certain time intervals. Lastly, the observations are summarized and the potential future work is discussed in Chapter 8.

2

Background

This chapter gives an overview of some of the fundamental concepts, taxonomy and hardware usage in the tactical domain. This will lay the foundation for understanding the literature work and methodology of this research study. Starting with a brief description of tactical networks followed by a basic three-layer architecture and Consultation, Command, and Control (C3) taxonomy [11]. Each of the layers of the C3 taxonomy in briefly explained with their functionalities and use-cases supported with examples.

2.1 General Architecture

Computer network enables the connection of a set of devices to communicate via applications. Any application serves a dedicated purpose such as exchange of messages, accessing resources, information collaboration, and so on. In general, the end-to-end process of an application can be categorized into three basic layers, namely the users, system, and network. This is shown in the general architecture (left) of Figure 2.1. Note that this model is how we visualize and categorize any system into the three basic layers. The user utilizes the service provided by the application by interacting with an interface. The application is deployed on the system which converts the user-message or requirements in the form of packets/signals that can travel through the underlying network technology. Therefore the system connects the users and network, however, in general, it keeps both of them opaque to each other. This means that users are unaware of the underlying network conditions and vice-versa. This three-tier architecture is applicable in several domains of computer science. One of the widely used examples is mobile phone, which involves a user, hand-held device that supports numerous applications serving varied purposes and the underlying cellular network represented on the right of Figure 2.1.

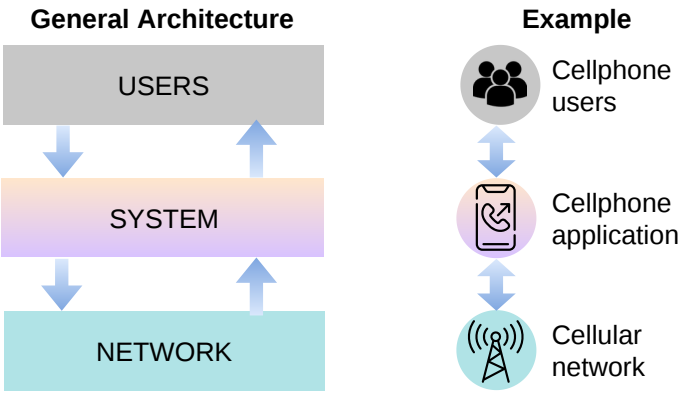

Figure 2.1 Interaction Model

2.2 Tactical Networks

In certain domains, the purpose of communication is not just to overcome the geographical barrier, but also to carry out domain-specific actions. This study concentrates on one such domain called Tactical Networks, which is a challenging field of application of computer networks. Tactical networks [43] support military communications that involve the exchange of information among military tactical force to enable Command and Control (C2). It is of high importance because of the communication of critical information to fight against terrorism, border tension, disasters, warfare, medical evacuation, the conduct of combat, and so on. Therefore, for organizing and collaborating operations, the armed forces rely on capacities of the tactical resources. In other words, military communication demands a reliable, secure, and timely transmission of information. Having understood the implications of tactical networks, the next section discusses more on the features and semantics of the tactical systems using one of the commonly used taxonomies in the tactical domain.

2.3 C3 Taxonomy

The framework of the three-layer architecture in Figure 2.3 can be extrapolated into Consultation, Command, and Control (C3) taxonomy which is widely used in the tactical domain. This taxonomy was built by North Atlantic Treaty Organization (NATO), which is an inter-governmental military cooperation that provides defense for North-American and European countries. This is represented by Figure 2.2. C3 taxonomy provides a shared platform for collaborating and coordinating ideas and strategies that are required to carry out military-specific activities and practices. This poses a subsequent problem of inter-operability which can be solved by introducing services, wherein a request invoked by the requester is catered by the service provider. This model consists of several layers which are dependent and related

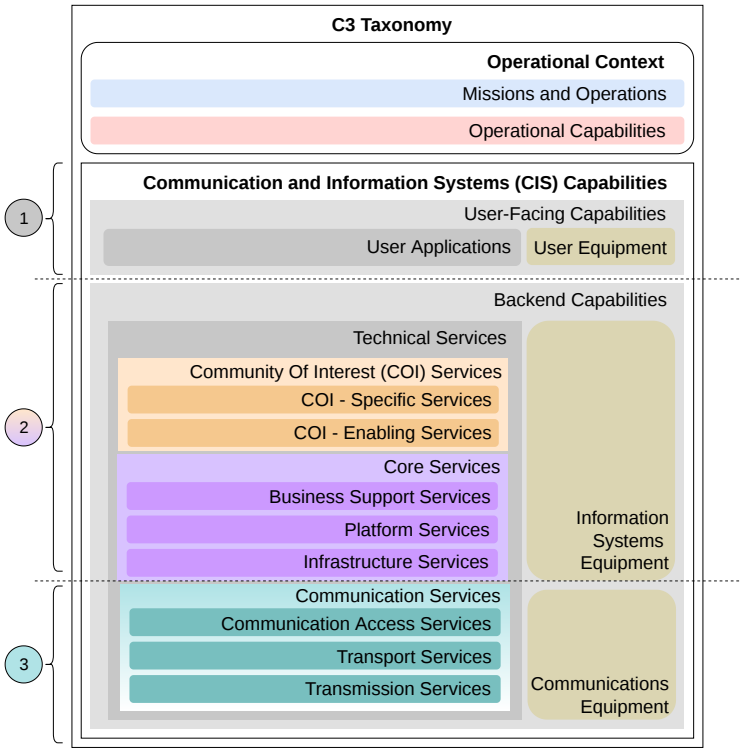

Figure 2.2 NATO's Consultation, Command, and Control (C3) Taxonomy referenced from [11]

to each other to function as a single block. The C3 taxonomy is broadly divided into two basic layers of *Operational Context* and *Communication and Information Systems (CIS) capabilities* [11].

The *Missions and Operations* layer represents the key values, rules, and approaches concerning organizing mission-oriented services. The *Operational Capabilities* specifies assignments and exercises to achieve the goal of accomplishing the mission.

Moving to the layer of concern is Communication and Information Systems (CIS), represents a combination of the software and the hardware requirements for carrying out the military-specific tasks.

2.4 Model Mapping and Fundamental Concepts

The following sub-sections discuss the functionalities of the different layers of C3 taxonomy that can be abstracted into the basic three-layer architecture. Before understanding the different layers of C3 taxonomy in detail, this taxonomy is mapped into the three-layer architecture as follows: user-facing applications and user equipment correspond highlighted as (1) in Figure 2.2 can be grouped into users layer, Community of Interest (COI) services, core services, and information systems equipment highlighted as (2) in Figure 2.2 can be amalgamated to tactical system layer and communication services along with communication equipment highlighted as (3) in Figure 2.2 correspond to the tactical network layer. This mapping is depicted in Figure 2.3 where the left column shows the basic three-layer architecture, the middle column shows the mapping of the C3 layers and the right column depicts some of

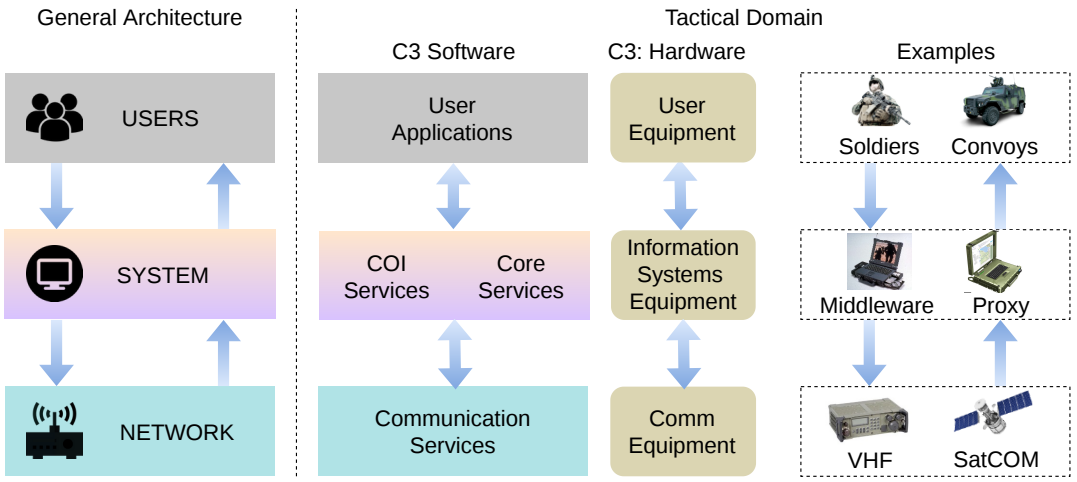

Figure 2.3 Model Mapping

the examples for each of these layers. Using this mapping analogy, the following sub-section discusses more on the software and the hardware components of the different layers of C3 architecture highlighting a few examples with a brief description of their working principle.

2.4.1 Tactical Domain: Users

The User-facing capabilities layer represents the communication of the users with the system comprising of the software component: *User Applications* and the hardware component: *User Equipment*. *User Applications* facilitates the users to perform a task by providing enough information on how to use the application. These user applications are accessed by military personnel, soldiers, convoys, the medical team as represented in the last column of the user application row in Figure 2.3. On the other hand, the *User Equipment* helps to run the applications on their devices.

Some of the examples of the user equipment are telephones, computers, laptops, tablets, and peripherals (I/O) units, radios, handheld devices, and many more. Enhanced Tactical Computers (ETC) are deployed on vehicles and are used to carry out military activities. On the contrary, in the recent past, there has been a shift towards the usage of a faster processor, lightweight, low cost and portable devices such as the hand-held devices, which enhance the visualization with a digital screen. They are used to send or receive classified data such as pictures, reports, or chat messages between military personnel. Recent developments include Blue Force Tracking (BFT) and some applications that provide a 360-degree view of the map and video communication capabilities.

2.4.2 Tactical Domain: Systems

The back-end layer of Figure 2.2 highlighted as (2), represents a collection of layers that has the implementations for handling the operational requests of the user applications. The *Technical Services* are broadly classified into *Community of Interest (COI) Services*, *Core Services*, and *Communication Services* [11]. The *COI*

Services support the collective tasks and assignments of the users. The *COI-Specific Services* facilitate the users in the realization of functional concepts for carrying out military procedures and practices and *COI-Enabling Services* caters to the interests of multiple user groups.

Moving to the next sub-layer, the *Core Services* layer helps to realize service-oriented approach supported by strategies that involve different frameworks. The *Business Support Services* collects and manages the information that can be leveraged by other components at different layers. *Platform Services* provides base support for assimilating services and technologies from different frameworks. Some of the common platform services are message-oriented middleware services and web platform services. The message-oriented middleware services facilitate the transfer of many message formats across different platforms and format. Web platform services facilitate the amalgamation of different services on a network platform that can be accessed by other components. The *Infrastructure Services* caters to providing a base for services in a shared system for carrying out military activities.

As seen, the functionalities of the COI and core services are commonly deployed using Service-Oriented Architecture (SOA), Web Services (WS), middlewares, and network proxies. This is discussed briefly as follows. Service-Oriented Architecture (SOA) is a design approach that provides the users to access the services with the help of an interface and network communication protocol. SOA is usually realized using web services [34] and through middlewares/brokers in tactical networks. Web services work on the basis of publish-subscribe mechanism where the service provider announces the services and maintains them in a repository which can be accessed by the consumer. Information exchange is facilitated by Simple Object Access Protocol messages.

Middleware provides services to applications and connects it to the operating system. Middleware works with the request-response process where the user interacts with the front-end to make a request. These requests are forwarded the back-end services which perform the necessary action. Recent studies have developed network-aware middleware to adjust the transmission rate [19, 10], shape the dataflow [30, 32], and efficient resource utilization [18]. Similar to a web service, middleware is responsible to hide the complexity of service through an interface that connects the front-end with the back-end. Network Proxy provides an exchange of information between two networks through gateways. Recent developments of NetProxy aim to improve QoS requirement, packet forwarding mechanisms, and packet fragmentation and defragmentation strategies [40].

2.4.3 Tactical Domain: Networks

The last layer of the *Technical Services* is the *Communication Services* which helps in the data transmission from the above layers into the network. *Communications Access Services* transfers data from core services into the transport layer. Some of the fundamental access services are exchanged in the form of analog, digital, message-based, packet, and frame. *Transport Services* mainly consists of services that facilitate transmitting data traffic across different topologies and networks. The *Transmission Services* constitutes the physical layer of communication of data

in the network. This is achieved using wired transmission services (Local Area Network (LAN), Metropolitan Area Network (MAN), and Wide Area Network (WAN)) and wireless Line of Sight (LOS) and Beyond Line of Sight (BLOS) transmission services [11]. Communication equipment is a device that facilitates data transfer between nodes. For example, routers, physical transmission media such as VHF radios, switches, network control equipment, satellite communication systems, and so on. The VHF radios enable long-range communication that is usually mounted on vehicles or carried as a manpack by the soldiers. The tactical IP routers are deployed in heterogeneous networks to facilitate information exchange, IP address configuration, forwarding data traffic, and enables routing protocol implementation [22].

The backbone of any communication is the underlying network technology. The three most widely used technologies are Very High Frequency (VHF), Ultra High Frequency (UHF), and Satellite Communications (SatCom) networks, which are typically used for long-distance data communication favoring military communication. VHF (30 - 300 MHz) is used in radio modems in tactical networks. UHF (300 MHz to 3 GHz) are widely used in the wireless communication with hand-held devices such as walkie-talkies. Satellite Communications (SatCom) facilitates the signal transfer between a source transmitter and a receiver at distant points on Earth with the help of a transponder, which augments the signals. All these communication technologies have nodes to facilitate transmission. These nodes can be static/mobile connected by wired/wireless network.

However tactical networks comprises of mobile nodes connected via wireless technology called Mobile Adhoc Network (MANET). It consists of a set of wireless mobile nodes that move randomly and rapidly with time causing topology changes. However, they are distributed, configuring nodes that lacks infrastructure [38], which makes them suitable for tactical networks. The wireless links have lower reliability with data delivery [46] and limited bandwidth [15]. This adversely affects the data transmission in the communication links.

Once the nodes are deployed, the next step is to determine if there is a valid connection to facilitate communication. In the literature, the network states are classified as *Disconnected, Intermittent and Limited*[47, 27]. Disconnection is the state of disruption in the network with no data transmission. Intermittent state has short interval of disconnection whose impact is quickly recoverable. For example: re-sending the lost packets. Limited state is defined when the link is hindered by low throughput, longer delay and higher packet error rates [23, 47]. In order to reduce the impact of long disruptions in the network, Disruption Tolerant Networks (DTN) is introduced [9]. DTN facilitates information exchange in the form of bundle transmission between the nodes [24]. The bundles are formed by grouping of packets.

3

Problem Statement

This Chapter outlines the problem statement that is planned to solve in this study. First, the problem is defined explaining the significance of solving it. Next, the context scenario and assumptions are discussed where the requirements of the network and the system layer in the ever-changing scenarios of tactical networks are explained. Lastly, the objective of solving the problem in this research study is discussed.

3.1 Problem statement

Tactical networks are prone to network heterogeneity, low bandwidth, high latency, and node mobility [40, 48, 7, 52, 18]. Also, they are subjected to frequent topology changes and disconnection, which characterizes the tactical networks to be volatile and *ever-changing* [55, 18, 30]. This causes hindrances in the data transmission process. For example, low bandwidth can result in data congestion and frequent disruption leads to packet loss. Therefore, there is a need for developing systems that can handle the network variation minimizing the impact on the data transmission process. In this regard, the problem statement of this research study is:

How to improve the robustness of the system to identify network variation/disconnection and subsequently adapt to network?

To mitigate this problem, several studies have developed suitable adaptation mechanisms to accomplish certain tasks such as modification of transmission rate [19] or prioritize the resource utilization [18] or transport protocol adaptation [41] or shaping of the data flow [33]. After developing any tactical system it should be tested rigorously against military network scenarios. However, testing with real military scenarios are expensive [13] and not easy to replicate [35] due to data confidentiality. In this sense, the testing process was narrowed down to using simulation/emulations in the laboratory environment. However, this process requires scenarios and data to replicate the tests in the laboratory [54, 51, 53]. Recent studies mitigate this

problem by acquiring real-time data and simulating the traffic into virtual platforms [13, 49], use Radio Frequency (RF) models [53, 54], and statistical distributions [45, 37, 32]. However, acquiring real-time data is still a challenging task and the RF models might not be applicable to all the radios as the configurations of the devices vary based on the manufacturers. Therefore, statistical distributions are used for the reasons of easy replication and no pre-requisite installation.

3.2 Context scenario

The next chapter discusses various architectures in which this study identified potential improvement to develop a suitable network adaptation mechanism in one of them [33]. The research began with the hypothesis that *the features of a store-and-forward mechanism such as the queuing model [33] can be leveraged to improve the robustness of the system to a) network changes by shaping the data flow based on network and system awareness and b) identify link disconnection.* This study leverages the use of favorable features of this architecture and also considers the constraints of the system that can be improved as a part of this research study. This architecture supports emulation with real hardware devices to understand the functioning and limitations of these devices. This is important in tactical networks because it has a high usage of hardware devices during mission-oriented tasks. Also, the limitations of this set-up such as the limited range of data rates supported by the hardware devices are taken into consideration while developing and testing the system.

3.3 Objective

The objective of this study is to improve the adaptation mechanism of the tactical system for network variation and disconnection. In this regard, the control mechanism is implemented to shape the data flow based on the dynamic parameters such as the network and system metrics. In addition, two link disconnection identification mechanisms are developed to improve network awareness in the system. This helps to reduce the impact of the problems caused on the system such as packet loss during link disconnection. The ever-changing network conditions are simulated in the laboratory to validate the robustness of the system.

4

Related Work

In the following chapter, the primary focus is to discuss some of the related studies which try to solve the problem discussed in the previous chapter (Chapter 3 *Problem Statement*). The literature review is classified into two sections compiling recent investigations on improving the robustness of tactical systems and investigations reporting experiments in different network scenarios. The first section discusses research studies that try to increase the system's robustness with varying network conditions. The second section describes the advantages and disadvantages of testing scenarios in the military and the laboratory environment. In addition, this chapter also explains some of the studies which try to build a testing environment in the laboratory.

4.1 System robustness

This section describes the implementation of a suitable adaptation mechanism for network variation and disconnection. In this regard, TACTICS' architecture is explained in detail because it is used as a foundation for this research study. In this process, the potential for improving the system's robustness to network changes is identified. The shortcomings identified in this architecture and other literature studies are used to design and implement a solution for the problem statement.

4.1.1 Adapting to changes in the network

The volatile tactical network behavior cannot be controlled, however, it can be addressed by reducing its impact on the services and applications. To solve this problem, recent studies have implemented mechanisms to detect network changes and adapt accordingly. They are briefly explained as follows.

Modification of transmission rates: In [19], the authors have identified that tactical networks consist of scenarios with data rates lower than the minimum necessary

to support the data outbursts from the applications. This can strain the application to ensure QoS requirements. In this case, the authors have proposed an adaptive middleware, that can sense the network conditions and adapt the transmission rates of the different applications based on their priority. This is implemented using three components, namely *Quality of Service (QoS) monitor*, *Adaptation Engine (AE)*, and *Control Interface (CI)*. The *QoS monitor* provides an overview of the data traffic by fetching the sent and the received bytes. This is sent to the *AE* which determines the changes in the data transmission rate for each of the applications based on the priorities. The changes in the transmission rate are executed using the *CI*.

Let us understand this mechanism with an example: let there be two data applications X and Y with priorities P_X and P_Y respectively, where $P_X > P_Y$. With the absence of the middleware, both of the data applications are allotted with the same bandwidth for execution. On the contrary, with the use of the middleware, when the available bandwidth is higher, the transmission rate of X is increased than Y, whereas when the available bandwidth is lower, the transmission rate of Y is decreased to make sure X gets the preference on utilizing the network.

The authors have improved their work in [44] to change the transmission rate based on admission requests sent to the *QoS agent* from the applications. The *QoS manager* visualizes the network conditions based on the sent and received bytes using a *Dynamic Throughput Graph (DTG)*. The transmission rates of the applications are modified by matching the application's request and viewing the available network conditions using *DTG*.

To improve the network awareness in the system, the authors in [10] have implemented *Network Awareness Service (NAS)* to fetch the network attributes, analyzing them, and circulating the information within the system. This mechanism can be used for efficient network utilization based on their availability. *Differential Loss Performance Analysis* mechanism identifies congestion based on the Packet Delivery Ratio (PDR). If the difference of the PDR of the higher and lower priority traffic crosses a pre-defined threshold, then congestion is identified. *Passive Measurement Analysis* tracks the packet traffic to fetch the network attributes. Based on the fetched attributes decision to reduce the transmission rate for efficient usage of the bandwidth. Throughput estimation is done by infusing packets into the traffic at specified intervals, called *Bandwidth Probing*. The throughput is estimated with time windows of 30 and 60 seconds [10] to observe smoother transitions. This research study implements and validates the throughput estimation with a time window of 30 seconds.

Transport protocol adaptation: The authors in [44] have implemented an adaptable transport protocol and improving the modification of transmission rate. The motivation for the former is that Transmission Control Protocol (TCP) is not suitable for the latency network because of longer recoveries and adjusting the traffic flow does not ensure certainty in case of congestion. For these shortcomings of the transport protocol, the authors have used *UDP-based Data Transfer (UDT)* protocol which is reliable, connection-oriented and at the same time provides a framework that can be configured to establish transparent control mechanism to change the transmission rate at the protocol level without being influenced with data congestion. In addition, it compiles the network characteristics such as sent and received

byte count and packet loss. The design principle of the cross-layer framework collects and distributes parameters throughout the system. It deploys a decentralized mechanism, wherein it is deployed on every node in the network to mitigate the network overhead and failures caused in the case of a centralized framework. The information gathered can be used for several adaptation mechanisms at different layers. In [41], the authors use this approach for transport protocol adaptation of TCP (*Derwood Lite*) in two steps. In the first step, the framework modifies the packet size based on the error rates. The TCP Gateway running on every node fragments large packets into smaller packets to reduce the margin of bit errors. TCP maintains a range of *Round Trip Time (RTT) and eligible rate estimation* values with normal and congestion network conditions. When the decision module senses, constrained network conditions (low bandwidth), it modifies the protocol window size and communicates it to the gateway to implement the decision. Also, the decision module also chooses the path with higher bandwidth for packet forwarding to efficiently use the available network conditions. The study concludes that the experiments conducted with cross-layer adaptation proved to be better than basic transport protocols with respect to packet delivery ratio [41].

Link classification: The goal of the study in [18] is to classify the link type using Agile Computing Middleware (ACM) technology. This technology consists of three components: *Network Sensor*, *Node Monitor*, and *Network Supervisor*. The *Network Sensor* passively collects the network details by extracting the packet information using Pcap/WinPcap library[1]. *Network Monitor* collects and circulates the network attributes to network supervisor. The *Network Supervisor* interprets the network conditions and classifies the link type into *HF/SatCom/LAN* link type. A set of pre-defined threshold values for throughput and latency stored in a configuration file for each of these links. During the execution of the experiment, the current network throughput and latency are retrieved by *Network Sensor*. The classification is done based on maximum confidence parameter obtained from the combination of stored and present network conditions.

Resource awareness and allocation: In [42], the authors have developed a priority-based resource allocation mechanism to efficiently use the common/shared resources. The approach leverages the use of *Network Management System (NMS)* to fetch the network characteristics using Simple Network Management Protocol (SNMP) protocol. The *Resource Negotiator* regulates the allocation of resources based on the fetched network attributes and the priorities of the applications. The results of resource allocation decisions is stored in the database which are then used by *Resource Allocator* to perform the required action. The *Resource Negotiator* compiles its findings whenever there are changes in the network. However, in the experiments conducted in the study, when the network parameters such as *delay* > 5 *and packet loss* > 20, asynchronous messages are generated to recompile the decision making process by the negotiator and corresponding allocation of resources to the higher priority applications are performed.

The ACM technology is extended by the authors in [16] and [17] to develop a network sensing methodology. In [16], they use this approach to reduce the overhead added to the network while circulating the networking information between nodes. In this sense, *Sensor Aggregation Service* combines data fetched from different sensors and transmits only a part of the vital network information to avoid overhead. In [17],

the authors use the same approach for increasing resource awareness in the system. Here, *S2ES* provides information about the status of the network at regular intervals to other middleware elements. This framework is tested in the *Anglova* scenario or the real-military test-bed wherein they highlight the shortcomings of confined knowledge and low-observability of the radios used in the experiments. This problem makes it difficult for independent scientists to reproduce the results for comparison or verification. On the contrary, the architecture of the Tactical Service-Oriented Architecture (TACTICS) project (used in this study) provides visibility to the radios in the network which is explained below.

4.1.2 TACTICS: Architecture

Cross-layer shaping of user dataflows: Tactical Service-Oriented Architecture (TACTICS) is an on-going research project developing a middleware that tries to provide solutions to mitigate data congestion by implementing a suitable store-and-forward mechanism [12, 26, 33]. This project is funded by *Bundeswehr*, through *Bundesamt für Ausrüstung, Informationstechnik und Nutzung der Bundeswehr (BAAINBw)* and *Wehrtechnische Dienststelle für Informationstechnologie und Elektronik (WTD-81)*, which are the unified armed forces of Germany. The architecture of this project is explored, followed by the solutions implemented to improve the robustness of the system.

Decentralization: Centralized approaches are not suitable for tactical networks as they introduce the risk of failures and considerable overhead for gathering and distribution of information through the network [41]. Therefore, the features such as neighbor discovery, packet fragmentation/de-fragmentation, routing and forwarding the data traffic are all implemented in every system realizing a decentralized stand-alone framework.

Hierarchical Queuing: The left-hand side of Figure 4.1 depicts NATO's C3 taxonomy [11], explained in detail earlier in Chapter 2. The exchange of information between users is supported by the user applications which triggers the corresponding user services. These services call the proxy comprised of *COI services*. The proxy intercepts the *service(s) messages* which are converted to the *IP* packets by the Transport Services. These *IP packets* are then transferred to the network through the communication devices. This is complemented by the architecture of TACTICS in which the data generated by the user message (A), flows through the hierarchy of queues in the sequence of the message queue (1), packet queue (2), and radio buffer (3) [33].

The VHF radio used in this study has a limited capacity of 128 KB for storing IP packets. The data is transferred from the radio buffer to the network (B) for transporting the data to the radio of the receiver using *User Datagram Protocol (UDP) protocol*. The information exchange within the system is facilitated by the cross-layer information exchange (4). The cross-layer information exchange is implemented using the contextual monitoring interface that helps to fetch the details such as queue size, occupancy, and threshold values from each of the layers and makes it accessible to other layers to implement system awareness. The neighbor discovery is done using the pro-active routing protocol Optimized Link State Routing (OLSR)v2 compiles its findings of the network graph details into a routing table [5, 36, 6].

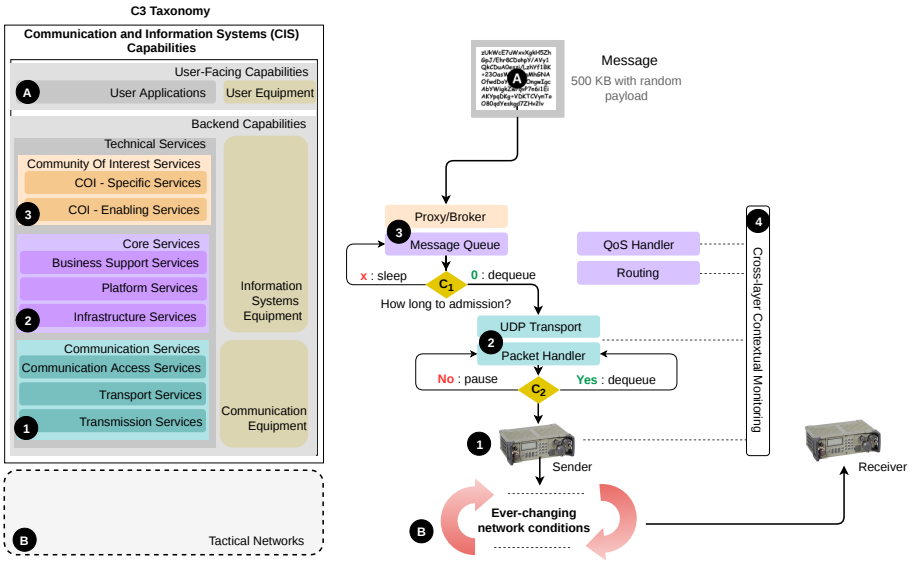

Figure 4.1 NATO's C3 Taxonomy [11] and Core Services [32]

Cross-Layer Information Exchange: As discussed, the design of cross-layer information exchange works on the principle of fetching the parameters and distributing to the different layers to increase the system awareness in this de-centralized approach. Figure 4.2 referenced from [30], illustrates the input(in), output(out), and control(co) chains deployed in the queuing model of every node in the network. Figure depicts three layers of the model starting with packet handler or $\{i_1, c_1, o_1\}$, message handler or $\{i_2, c_2, o_2\}$ and service mediator or $\{i_3, c_3, o_3\}$ [33]. Whenever a service triggers a request, it traverses through the system in a sequence of $\{i_1, i_2, i_3\}$ and corresponding reply in the opposite direction in a sequence of $\{o_3, o_2, o_1\}$. Also, at every layer, there is the control plane $\{c_1, c_2, c_3\}$ which shows that a suitable mechanism could be deployed in these layers based on the input/output received from different layers. For example, using this interface, the radio buffer details such as radio buffer occupancy and threshold can be accessed at different layers.

Control Points: To avoid data congestion, the store-and-forward mechanism implements two control points C_1 and C_2 (seen in Figure 4.1) between the three layers of the queuing model. On intercepting the messages from the proxy, the message

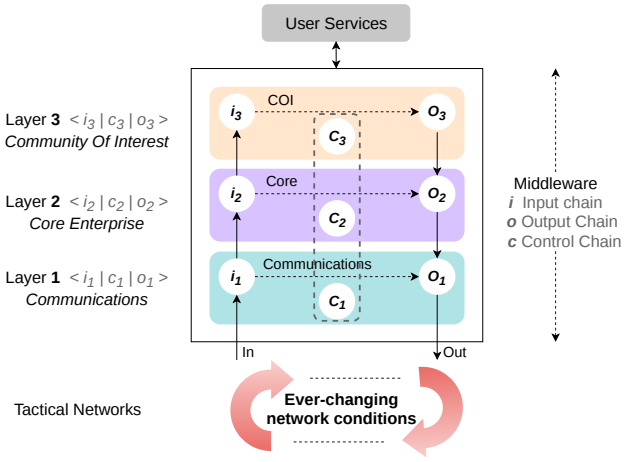

Figure 4.2 Tactical middleware in/out chains at a particular node [30]

queue Q_m transfers the messages to the packet queue. The control point C_1 denotes the time for admission determined by *QoS handler* using Equation 4.1 [33]. In this equation, Qp_0 is the default packet queue, Qp_r is packet queue, ΔB_r is the current buffer and Δout is the radio output.

$$Time\ for\ Admission = \frac{(Qp_0 + Qp_r + \Delta B_r)}{\Delta out_r} \tag{4.1}$$

The messages are retained in the message queue until Equation 4.1 at C_1 yields 0 seconds. Later, the messages are transferred to the packet queue. The UDP transport protocol fragments the IP packets in the packet handler. The transfer of packets from the packet handler into the radio buffer is regulated by the control point C_2. The absence of any control mechanism at C_2 results in flooding of the data packets (message outbursts) into the radio buffer exceeding its capacity (128KB), thereby resulting in the buffer overflow and packet loss. This is explained using Equation 4.2. In this Equation, *in* and *out* denotes the inflow and outflow of the packets from the buffer respectively, α_i and β_i are the flow variables (subjected to change over time) and ΔB is the buffer occupancy. The buffer overflow occurs when $\Delta B > 128KB$. However, this indicates an extreme case with the absence of any control point at C_2. Therefore, in [33], they have introduced the *Threshold Shaping* control mechanism. The radio threshold (b) refers to the static value stored in the configuration file which is the (%) of the total capacity of the radio ideally suited for storing in the buffer. The packet handler continuously monitors the radio buffer occupancy ΔB_r at recurring intervals (∼2seconds), and transfers packets to the radio buffer until the threshold is reached. Once, the occupancy exceeds the threshold, the packet handler pauses the transmission until the packet transfer from the sender radio to the receiver radio reduces the buffer occupancy to a value lesser than the threshold ($\Delta B < b$). As a result, the occupancy does not cross a predefined threshold value and thereby, the data loss due to buffer overflow is reduced.

$$\Delta B = (\alpha_i in_i - \beta_i out_i) \leq b \tag{4.2}$$

4.1.3 Disconnection identification and re-transmission

As already discussed, tactical networks are highly heterogeneous subjected to frequent disruptions [55, 18]. In such cases, it becomes identify link disconnection and reduce its impact on data transmission. The link disconnections are identified based on the traffic features such as delay/latency and loss using machine learning models [50], estimation functions [25], time interval based link breakage predictions depending on the power of the received signals [20]. Machine learning models are also used for network parameter estimations [8]. Disruption Tolerant Networks (DTN) is one of the methodologies used to cater to the network experiencing frequent disruptions [24, 39]. In [39], authors have implemented a DTN proxy to re-send the lost data packets during network disruptions. This is implemented in three steps. In the first step, *IP Packet Interceptor* receives the IP packets from the traffic and passes it to the *TCP proxy*. The *proxy* receives the packets, sends an acknowledgment, and combines a large number of packets (to reduce overhead) into a *bundle* and forwards

Studies	Goal	Mechanism
	Network Variation	
QAM [19]	Modification of transportation rates	*QoS monitor fetches the sent and received bytes*
QAM [44]		DTG projections to visualize the network traffic
NAS [10]		Passive and active monitoring to fetch network parameters such as throughput, delay
QAM [44]	Transport protocol adaptation	*Implementation of the UDT protocol*
DISCO [41]		MTU accommodation and link switch based on network parameters by *TCP Derwood Lite*
Auto-DRM [42]	Resource allocation and access	Collects and compiles network metrics from SNMP
SENSEI [16]		*S2ES* provides regular network reports to perform network analysis
DDAM [18]	Link classification	*Network Sensor* collects throughput and latency to calculate confidence scores for each link type
TACTICS [30][32]	Shaping of the data flow	Delay mechanism based on OLSR routing information feedback
	Network Disconnection	
DTN [39]	Link classification	*DTN transport for transmission of bundles between proxies*
MaNaTIM [7]	Retrieve lost data	Data integration and coordination
[50, 8]	Link estimation and fault identification	Machine learning models
[20, 25]	Link disconnection identification	Network and signal parameters

Table 4.1 Adaptive middlewares for ever-changing network

them to the remote proxy. The *DTN Transport* is required for *bundle* transmission between the proxies. Transmission between proxies consists of a number of routers that exchange the *bundles* with each other through a TCP connection. The advantage of having multiple connections is that, in case of disconnection, the lost *bundles* are re-sent from the previous router instead of traversing the whole link again. The predecessor proxy to the destination, de-fragments the bundles into respective packets, and delivers them to the destination.

Moreover, there are shortcomings in deploying DTN techniques to tactical networks [4]. These consist of directing non-DTN data traffic to pass through a DTN proxy and enhance the already in-use technologies to accommodate DTN. The authors in [7], have deployed a middleware to amalgamate the functioning of tactical networks with business applications. *Information Management Services Bridge (IMSBridge)* maintains a data repository of all the information exchanged. This feature is leveraged for data integration and coordination in case of disruptions. While carrying out this process, available network conditions are considered. For example, in the case of constrained network conditions, only the current updated information is retrieved and vice-versa.

Table 4.1 summarizes the adaptation of the middlewares to the varying network variation and disconnection. The first column provides the list of research studies, the second column specifies the goal for adapting to the network conditions followed by the last column that specifies the mechanism of retrieving the network parameters. Although all the above-mentioned studies work towards the same goal of increasing the system's robustness to changes/disconnections in the network, however, the sensitivity to detect network changes is not examined. Moreover, the system needs to be tested with ever-changing network conditions and leverage the available system and network parameters to improve the robustness of the system. The next step is to look at the importance of testing the system with real military and simulation scenarios.

4.2 Experiments with different network scenarios

Tactical networks have certain shortcomings such as network heterogeneity, low bandwidth, high latency, and node mobility [40, 48, 7, 52, 18]. In addition, they are subjected to frequent topology changes and disconnection, which characterizes the tactical networks to be volatile and *ever-changing* [55, 18, 30]. This causes irregularity and unpredictability in the network conditions. Any adverse effect on the network characteristics causes a significant impact on the data transmission which should not be expected in case of an emergency like warfare or medical evacuation. Therefore, it is important to test the tactical system rigorously with the military network conditions.

4.2.1 Difficulties of testing in real tactical environment

Testing of the tactical systems in real military scenarios needs technical developers who can quickly identify and correct the issues faced by the system. However, this process is not cost-effective [13]. Moreover, the on-field testing process cannot be replicated to analyze the results [53]. In other words, the test data is difficult to access because of the norms of confidentially. For example, in the research study of [35], the authors have implemented a transparent proxy to decrease the latency in information exchange via e-mail and Blue Force Tracking (BFT). The authors claim that the results were successful for all the test cases as the information was delivered even when the network was prone to long delays. The authors also stress that even though the testing was done in a real-military scenario, the experiments were not conducted thoroughly (for testing QoS improvements) because it required more time which contradicted the compact schedules. Moreover, network conditions such as disruption occurrences/frequencies, bandwidth are not specified. Therefore, it is not only difficult to access the test data but, also difficult to replicate them in the laboratory environment.

4.2.2 Advantages of testing in simulated/emulated environment

Therefore, *Simulation* and *Emulation* have gained importance especially in the tactical context in recent years. The experiments conducted by simulations are inex-

pensive, can be replicated and the network parameters can be changed or regulated [53] based on the requirement. Moreover, this also helps in the early identification of defects of the system that can be rectified before their actual deployment. However, they need scenarios to conduct experiments [54, 51, 53]. But the problem is that even though many scenarios can be found, they do not provide all the adequate details to replicate experiments. As discussed before, this is because the military test case details are not easily accessible, which limits research investigations to use a common ground to measure and analyze the results [54]. The following research studies are classified based on the mechanisms they use to test in an emulation/simulation platform. The mechanisms are i) introducing real-time data into virtualized hardware devices, ii) use *Radio Frequency (RF)* and waveform models and iii) statistical distributions.

Real-time data into virtualized hardware devices: The study in [13] highlights that conducting experiments in real military scenarios is expensive and therefore, suggests an alternate approach in the laboratory set up. *JTRS Enterprise Network Manager (JENM)* is used to collect and store real-time data traffic and radio parameters. *Software Virtual Networks (SVN)* is used to feed this data into the network making sure that it is no different than the live traffic from the application perspective. *Joint Network Emulator (JNE)* interface intercepts the traffic and forwards it to the virtual radio whose movement is managed by *One Semi-automated Force (OneSAF)*. Hence, experiments are conducted with real-time data induced into the virtual environment to conduct operational tests in a laboratory before actual deployment.

In [49], the study is about creating a simulation environment for training purposes and uses Optimized Network Engineering Tool (OPNET) for implementation. This approach is inexpensive and allows the simulation of hardware devices into virtualization units that can be adjusted based on the requirements. This is used for training the military personnel to perform activities and practices. To create a military scenario, it depends on the input file that is further converted into parameters that can be used to conduct simulation experiments. The in-built virtualization models can be used and the network parameters are calculated in the background when any changes are done to the models using the simulator. The calculated values are then used for determining the packet transmission. Even though in the above experiments they introduce real-time data into the virtual platform, the challenge of collecting this data persists, and using virtual radios/devices limits the observation of the shortcomings on the hardware devices.

Radio Frequency (RF) waveform models: In [53], the authors have deployed an *Extendable Mobile Ad-hoc Network Simulator (EMANE)* emulator for the top three layers of the Open Systems Interconnection (OSI) stack. The authors have developed different waveform models applicable to *Radio Frequency (RF)* levels to vary the network parameters such as the data rate, latency, and also node position and movement. The experiments conducted are similar to actual military scenarios such as information collection and medical evacuation.

In [54] the authors discuss that testing the system with real-military set-up helps to evaluate the systems but the test cases and data are not readily available for replication. Therefore, the authors have used *Extendable Mobile Ad-hoc Network Simulator (EMANE) emulator* to test their system. The study discusses a few ex-

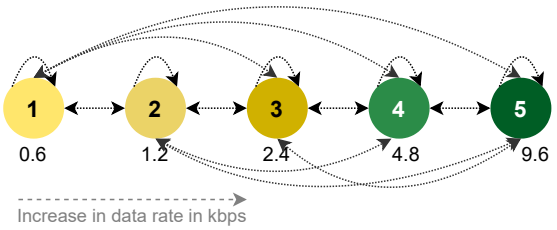

Figure 4.3 Data rate network states [32]

periments to analyze the performance of the OLSR routing protocol with various *Narrow Band (NB)* and *Wide Band (WB)* waveforms using *EMANE emulator*. The quality estimation of the routing protocol is analyzed by comparing the link quality to a threshold value. The study also highlights the shortcomings of using this emulator such as the need to run a single instance to maintain time synchronization, the disparity in the sender and receiver data rate modulation, and packet loss due to limited buffer capacity [54]. The waveform propagation models might not be applicable to all the radio hardware devices as the configurations vary depending on the manufacturing company. In addition, the emulation environment should be common for a fair comparison of the systems, which obligates the additional effort of installation and deployment of the emulators in the framework.

Statistical Distributions: The research study in [45] was developed to conduct extensive experiments in the laboratory set-up. This approach leverages the use of cross-layer information exchange to distribute the network parameters compiled by the Optimized Link State Routing (OLSR) routing protocol to increase network awareness in the system. *simTAKE* regulates the packet flow at each node by changing the network attributes, for example, by increasing/decreasing the throughput in the network, latency. On the other hand, to simulate the user data generation, *emulTAKE* is used to inject messages into the traffic based on statistical distributions.

In [37], the authors have built a tool to carry out military activities in a simulated environment in two steps. In the first step, they build simple military models catering to ranking and grouping of objects comprising of tactical units, personnel, and hardware devices. These models can be replicated to meet scalability. In the second step, the network emulation is performed by assimilating the functionality of the tool with *Common Open Research Emulator (CORE)* [3, 2] and *SimPY* technology. Using this combined technology, they are able to emulate the distinct task-based military activities whose time interval is based on statistical models such as the *Gaussian distribution*. The network attributes such as the delay and capacity are set before the start of the experiments.

In [32], the author's goal is to introduce randomness while creating network conditions in the laboratory environment. Therefore, they have used Markov model to generate network state patterns wherein the states correspond to the data rates configured in the radio namely 0.6, 1.2, 2.4, 4.8, and 9.6 kbps as shown in Figure 4.3 [32]. The input to the Markov chain model is the data rate states from Figure 4.3 whereas the output is the sequence of network states.

To emulate the data rate sequence in the network, a python script is implemented in each of the nodes which takes the input as the data rate sequences and correspondingly changes the data rates in the radio by modulating the waveform on run-time

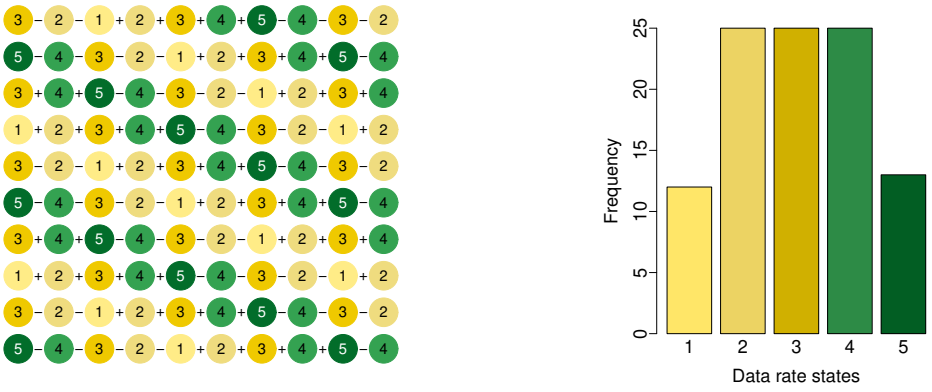

Figure 4.4 D_2 data rate sequence (left) and frequency (right) [32]

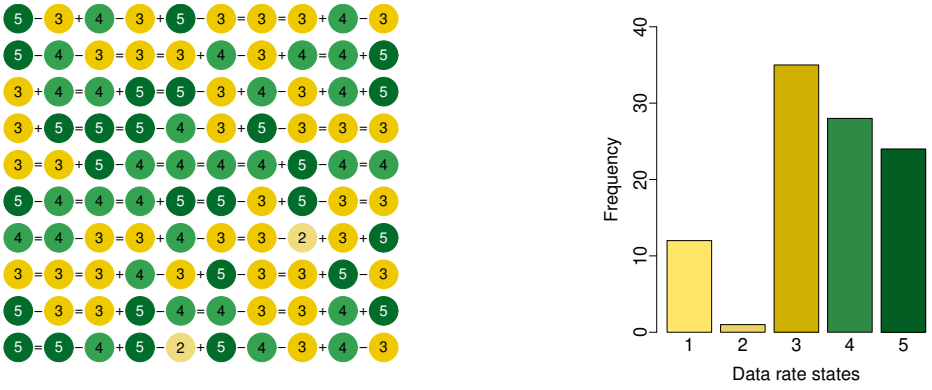

Figure 4.5 D_3 data rate sequence (left) and frequency (right) [32]

for the specified time intervals. Instantiating this model, three data rate sequences were produced. Of these three sequences, two of them are discussed as shown in Figure 4.4 and Figure 4.5. In both the figures, the data rate pattern is depicted on the left and the frequency of occurrence of states is shown on the right. Both the data rate sequences traverse from left to right along every row from bottom to top. The numbers inside the state depict their corresponding data rates as seen in Figure 4.3. In addition, the signs +/-/= indicates an increase/decrease/same data rate transition. D_2 data rate pattern follows a pendulum pattern with 25% frequency of occurrence for states 2,3 and 4 while states 1 and 5 have a frequency of 12% and 13% respectively. D_3 data rate pattern has a higher frequency of occurrence for states 3,4 and 5 with values 35%, 28%, and 24% whereas states 1 and 2 have a frequency of 12% and 1% respectively. By implementing the data rate sequences in the radios using the script, the robustness of the middleware to control the inflow of data to the buffer for network variation is examined.

This research study stands distinct from the aforementioned simulators for the following reasons: a) combines the waveform propagation to change the data rate of the radios and statistical distribution to introduce randomness in the network conditions b) usage of statistical distributions makes it easy for replication and scalability for other network parameters c) usage of actual hardware set-up rather than in a simulation environment (easy to observe the limitations) and d) does not require any prerequisite installation of the simulators/emulators thereby hindering the shortcomings of their usage. However, these testing scenarios might not match the

Testing scenarios		
Scenario	**Advantages**	**Disadvantages**
Real military scenario	Best form of validation Able to identify and quickly rectify defects	Expensive Difficult to replicate
Simulation/Emulation	Cost effective Rigorous testing before deployment	Require scenarios to be replicated Virtualization - difficult to observe hardware limitations

Table 4.2 Advantages and disadvantages of testing scenarios

Simulation/Emulation environment		
Studies	**Goal**	**Mechanism**
[13]	Test the application to identify defects before deployment	Real-time data into virtualized hardware devices
[49]	creating a simulation environment for training purpose	
[53]	Test military scenarios such as information collection and medical evacuation	*Radio Frequency (RF) waveform models*
[54]	Testing the reliability of *OLSR protocol*	
[45]	Emulate message into the network to observer the system's performance	Statistical distributions
[37]	tool to carry out military activities in a simulated environment	
[32]	Introduce randomness while creating network conditions in the laboratory environment	

Table 4.3 Adaptive middlewares for ever-changing network

fidelity of real tactical scenarios. It is assumed that using these statistical distributions one can produce sequences simulating a wide range of scenarios (including best and worst cases) which will help us to understand the performance bounds of the system and provide a common baseline for quantitative comparison for validating the systems. Therefore, in this study the statistical distributions are used to create network variation.

The occurrence of disruptions is frequent in tactical networks. However, the replication/simulating disruptions in the studies are missing. Therefore, there is a need to induce link disconnection to test if the system is robust to identify and subsequently handle the repercussions.

Table 4.2 highlights the advantages and disadvantages of testing the tactical system with real-time military scenarios and the simulation/emulation scenarios. Table 4.3 is divided into three columns where the first column shows the studies conducted, the second column highlights the goals of the research and the last column explains the mechanism used to create the testing environment.

5

Design

The previous Chapter 4 (Section 4.1.2) explains the architecture of Tactical Service-Oriented Architecture (TACTICS) project which implements a hierarchical queuing model of the message, packet and radio buffer in a middleware [33]. This middleware is used to solve the research question of *"How to improve the robustness of the system to identify network variation/disconnection and subsequently adapt to the network?"* The distinct features of this hierarchical queuing model are cross-layer information exchange and interface to the radio. Keeping this in mind, a solution was proposed for the problem of improving the robustness of the system to varying network conditions and disconnections that can be tested with TACTICS architecture. In that way, the research began by exploring the system's sensitivity to network variation and disconnection. From initial experiments, it is observed that i) the network awareness can be used to shape/regulate the data flow, and ii) system cannot detect network disconnection and suffers from packet loss. Based on initial observations and the potential to improve this architecture, the goals of the research study are derived.

Figure (5.1) shows that the objective of the study is to design a system and subsequently test it in the laboratory environment. The experiments are conducted in the laboratory environment because of the shortcomings of testing in real-military scenarios highlighted in the literature study. The study proceeded by designing two scenarios for testing the robustness of the system to network variation and disconnection. For the former, S_A, data rate was used as the network parameter to vary over time because data rate has a direct correlation with the data transfer, for example, higher the data rate, more packets are sent/received in the network, and vice-versa. For the latter, S_B, the link was disconnected using the hardware prototype. For each of these scenarios, suitable mechanisms are developed to identify and subsequently adapt to the network conditions. These mechanisms are evaluated by the definition of a model using real radios in the laboratory. For each of the identified scenarios, adaptation mechanisms are implemented and subsequently tested as seen in Figure 5.1. For S_A, a dynamic data flow adaptation is designed which comprises of data rate computation (1) and introduction of a delay called Inter-Packet-Interval

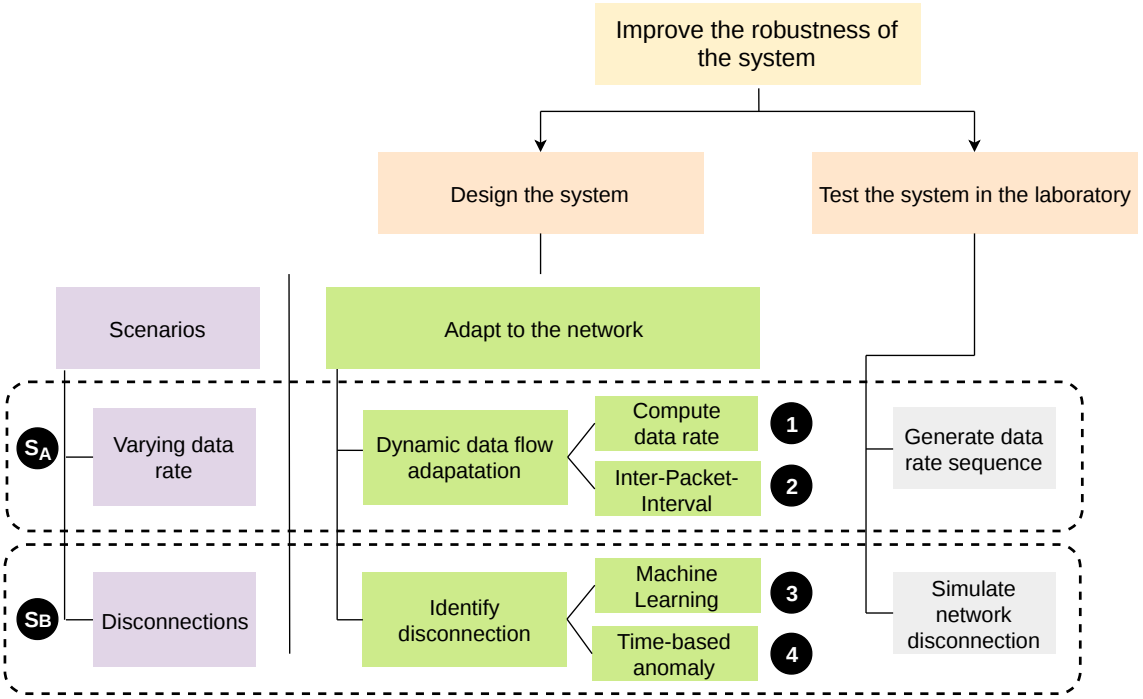

Figure 5.1 Design Overview

between packet transmission (2). This mechanism is validated by simulating the link with data rate sequences. On the other hand, for S_B, machine learning approach (3) and time-based anomaly model (4) are used to identify link disconnection. This mechanism is validated by simulating the link disconnection.

5.1 Dynamic adaptation of dataflows

The store-and-forward mechanism in TACTICS architecture implements a mechanism to adjust the flow of data through the pipeline depending on the threshold [33]. However, the shaping of the data flow is influenced by the ever-changing network conditions. To make this control mechanism more robust to network changes, it is implied that the network parameters need to be taken into account. The network parameters are the link data rate, packet loss percentage, Signal to Noise Ratio (SNR), Received Signal Strength Indicator (RSSI), and many others. However, link data rate was used for further analysis because i) this can be retrieved from already deployed protocols in TACTICS framework, and ii) fetching of other parameters requires the installation of active tools which can increase network overhead.

The link data rate is retrieved from the Optimized Link State Routing (OLSR) routing protocol and Simple Network Management Protocol (SNMP) protocol. The former uses an active probing mechanism that uses the link to compute the capacity, whereas the latter is a passive approach that measures the current data flow and checks the radio modulation defining the maximal nominal data rate. An experiment is performed to analyze the sensitivity of both the protocols to data rate changes in the network. This is illustrated in figure 5.2 consisting of two plots. The first plot (above) has OLSR (red) and SNMP (grey) compiled link data rates. The second plot (below) has the data rate simulated in the network varying from 0.6 to 9.6

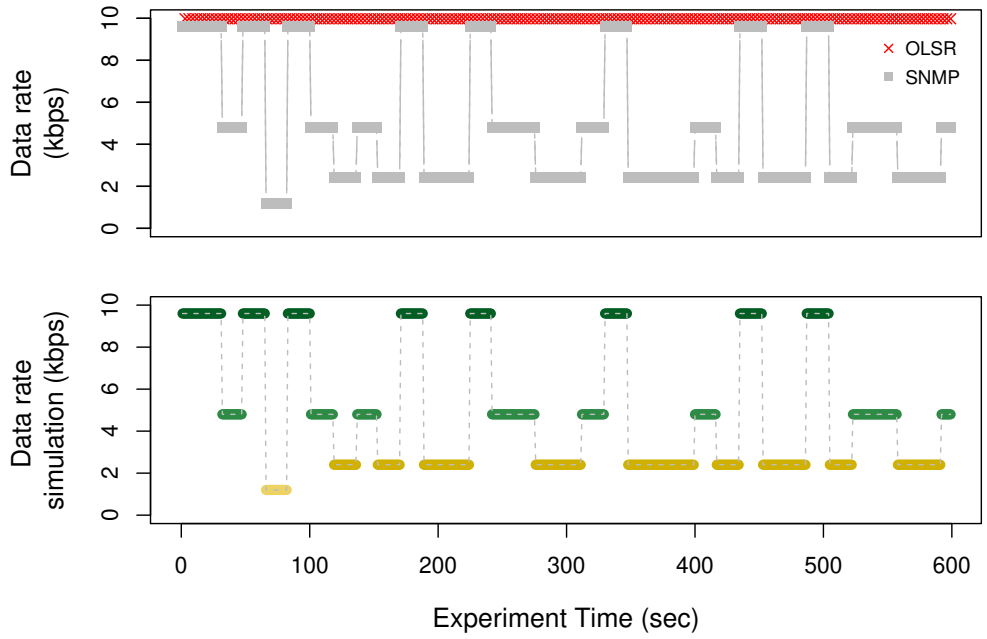

Figure 5.2 Sensitivity comparison of protocols for data rate changes

kbps following a random sequence. It is visible from the figure that SNMP is more sensitive than OLSR to detect data rate changes in the network. To improve the sensitivity of the latter, *hello interval* of the routing protocol was decreased but, this added overhead in the network hampering the data transmission process. Since adding overhead is unfavorable in the given framework, SNMP is used to retrieve data rates for further analysis.

5.1.1 Link data rate computation

In absence of any mechanism to obtain the data rate from the network, cross-layer information exchange is leveraged to compute the data rate in an offline model without adding any or imperceptible additional overhead on the system. This is an important point of consideration in low bandwidth networks.

Figure 5.3 depicts the below layer of the hierarchical queuing model with the radio of the sender and the receiver on the left and right, respectively. The sender radio receives the IP packets from the packet handler which flows through the radio and link in sequence before reaching the receiver. At a given point in time, there is access to the data flow of the IP packets sent to the radio and the radio buffer occupancy. Using these metrics within a time-window the number of IP packets leaving the buffer can be computed. It has a direct correlation to the link data rate. For example, the higher the link data rate, more the number of packets are transferred from the radio buffer to the network medium. Based on that, an experiment is conducted by varying data rates in the network. This is illustrated using Figure 5.4 where the data rate simulation in the network is shown in the first plot (from below) ranging from 0.6 to 9.6 kbps. The second plot (from below) shows the number of IP packets intercepted at the receiver whereas, the third plot shows the number of IP packets

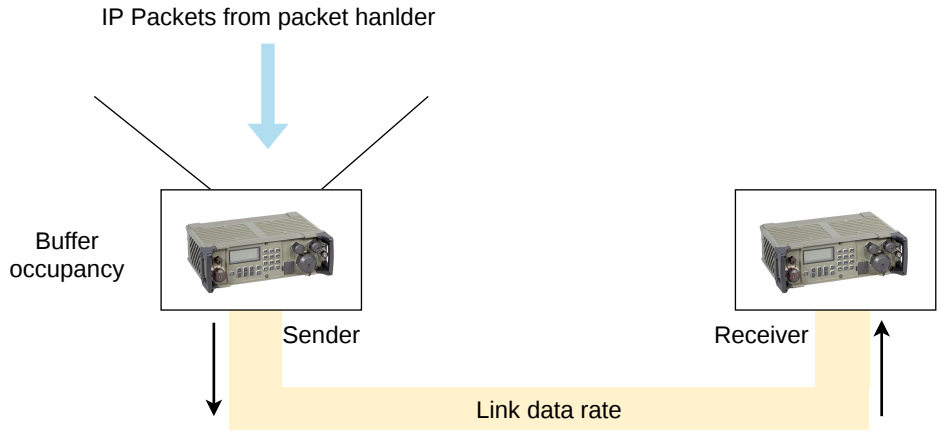

Figure 5.3 Network setup and metrics for data rate computation

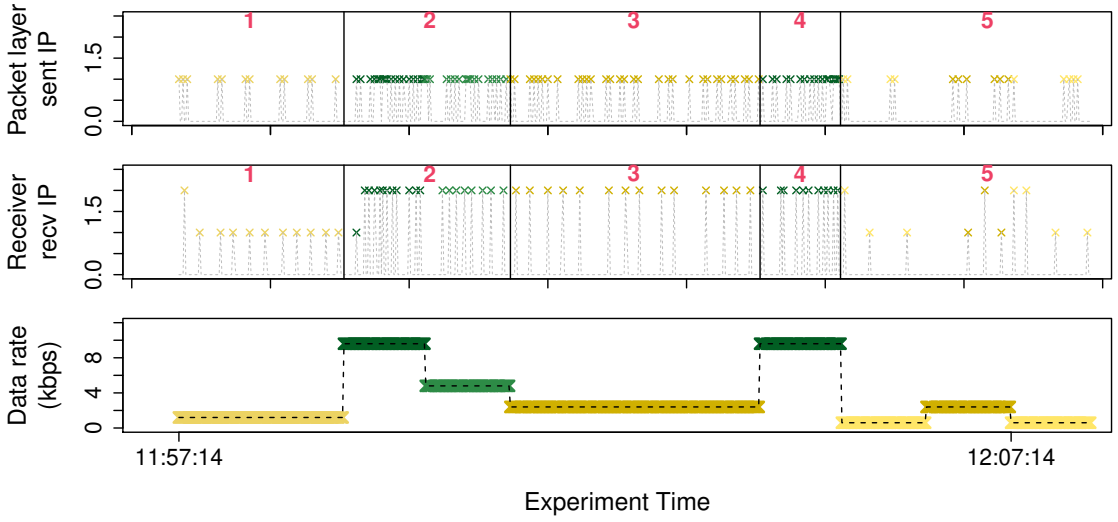

Figure 5.4 Rate of IP packet transfer for varying data rates

transmitted at the packet layer. The second and third plot is divided into five regions (1,2,3,4,5) highlighted to show the behavior of the sender (packet layer) and receiver for different data rates. The first (1), third (3) and fifth (5) intervals have a lower data rate of (0.6-2.4) kbps, in which the frequency of the data packets transmitted at the packet layer is comparatively lower. Likewise, the frequency and number of the packets intercepted at the receiver are also lower(= 1). On the contrary, the second (2) and fourth (4) intervals have a higher data rate of 4.8 to 9.6 kbps which makes the frequency of packet transmission and interception higher at the sender(packet layer) and receiver respectively. Moreover, the number of packets intercepted at the receiver is comparatively higher (= 2).

Using this analogy, the link data rate at the sender is computed without considering any parameters from the receiver side. This mechanism is highlighted in Figure 5.1 as (1). However, this methodology might not be applicable in case of the implementation of passive tools or disconnections in the network. In the latter, the system does not recognize the link disruption and still continues to send packets from the buffer which are eventually lost, in which case the computed data rate shoots above the maximal capacity of the link.

5.1.2 Inter-Packet-Interval

The objective of having a control mechanism at the packet handler is to avoid the data overflow in the buffer, therefore, Internet-Packet-Interval (IPI) is proposed to introduce a delay mechanism. This is explained using Figure 5.5 which shows the hierarchical queuing model to highlight the design of the control mechanism. The need to regulate/shape the inflow of data into the radio buffer is a continuous process and requires an active control loop that determines the delay time interval, called Internet-Packet-Interval (IPI) for every packet transmission from the packet queue to the radio buffer at C_2. This mechanism takes input from the below two layers namely the radio buffer (buffer metrics) and the network (link data rate). The former is required to understand the available capacity in the buffer while the latter helps to understand the rate at which the already existing data in the buffer is transmitted to the receiver. To benefit this approach, cross-layer information exchange fetches the parameters of the subsequent/next layers in the pipeline and distributes them across the system represented by $<in>$ *parameters*. Access to the radio buffer metrics (threshold and occupancy) and link data rate from SNMP is provided by cross-layer information exchange at C_2 shown in the figure as the input parameters to IPI. The output of this control mechanism decides if the packets should be de-queued to the radio buffer or not. In the case of the former, a delay is introduced between packet transmission (from packet handler to radio) while in case of the latter, the mechanism pauses the transmission until there is space in the buffer to be filled. If however, in the absence of access to data rate compiled by either of the above-mentioned protocols, the mechanism can use the online methodology of section 5.1.1 to compute the link data rate.

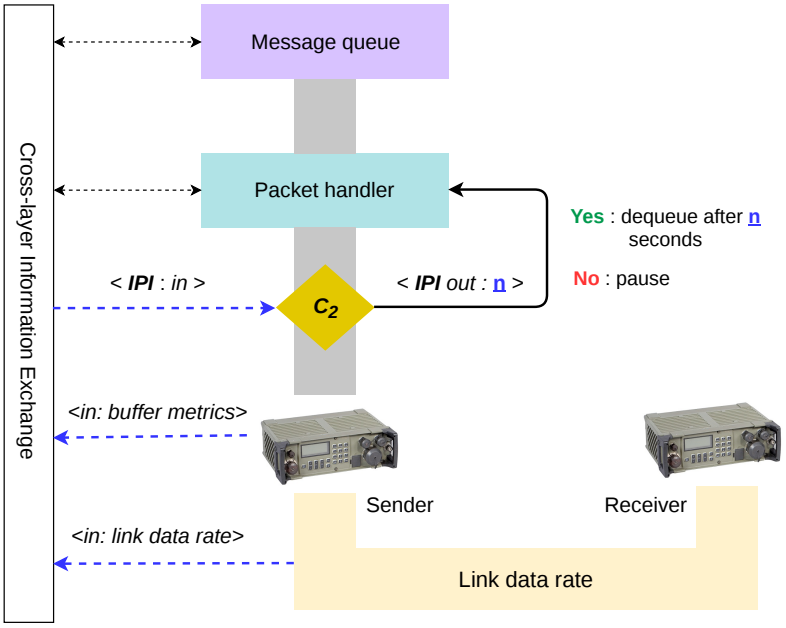

Figure 5.5 Design for Inter-Packet-Interval

This mechanism is highlighted in Figure 5.1 as (2). In this sense, to maintain the inflow of the data to the radio buffer, a control mechanism is implemented at packet handler based on: (i) the data rate in the network and (ii) buffer metrics: threshold and occupancy. More knowledge about the different layers improves the robustness

of the system with the dynamic adaptation of the data flow to varying network conditions.

5.2 Detect network disconnection

The research in [32] tests the system's response to network variation and uses Markov chain model to simulate the data rate sequences in the network. However, this approach does not consider link disconnection, which frequently occurs in tactical networks. This is considered a potential improvement that can be done to improve the system's response mechanism. With this idea, the study is classified into simulating disconnection in the laboratory and implementing a suitable mechanism to identify the link disconnection. For both the former and latter, the different steps taken during the process are described highlighting the shortcomings and alternative approaches that fit the requirement and scenario.

5.2.1 Link disconnection

The pre-requisite for testing of system's response to disconnection was to simulate the link disconnection. Initially, software tools/commands were used to cause link disconnection which affected the routing protocol causing the entries of the routing table to disappear even when the connection was re-established and resulted in very large recovery times. In addition, the various services running on the middleware had to be restarted with every disconnection. The next alternative was to simulate this process with the help of a hardware device. A prototype was designed with Raspberry-Pi because it was cost-effective, has multiple input/output pins and ease of deployment. However, it needed the support of other hardware devices such as a relay, adjustable converter module, and an attenuator to complete the circuit for causing link disconnection. Notice that channel emulators could also emulate the disconnection, however, these equipment are expensive, motivating the development of a low-cost approach. The disruption constitutes as a valid network state along with other states representing the data rates. The already existing network state model in [32] was improved to imbibe disconnection as shown in figure 5.6. The edges of the figure show the transition between the states. Next, this disconnection model is integrated with the hardware device. This requires the specification of the network sequence along with the time intervals for controlling the experiment. In that way, statistical distributions were used to create the sequences of network states and state transitions.

5.2.2 System behaviour to disconnection

An initial experiment was conducted by simulating two disruptions of 60 seconds each at the network layer. Figure 5.7 plots the behavior of the parameters at different layers during the experiment. The first and second (from above) row of this figure shows the packet queue occupancy (% B) and radio buffer occupancy (% KB) plotted over time. The third, fourth and the fifth rows indicate i) packets sent by the sender,

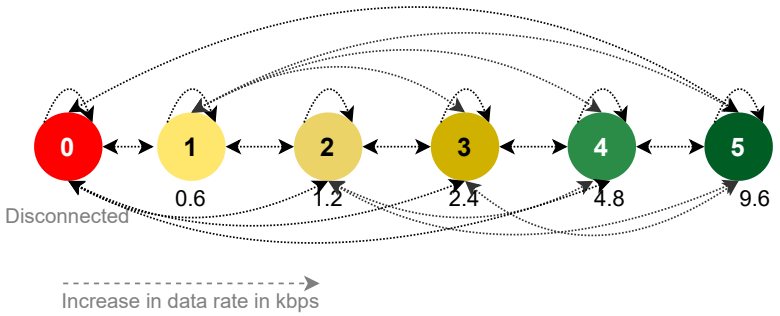

Figure 5.6 Network model with disconnection. Adapted from [32]

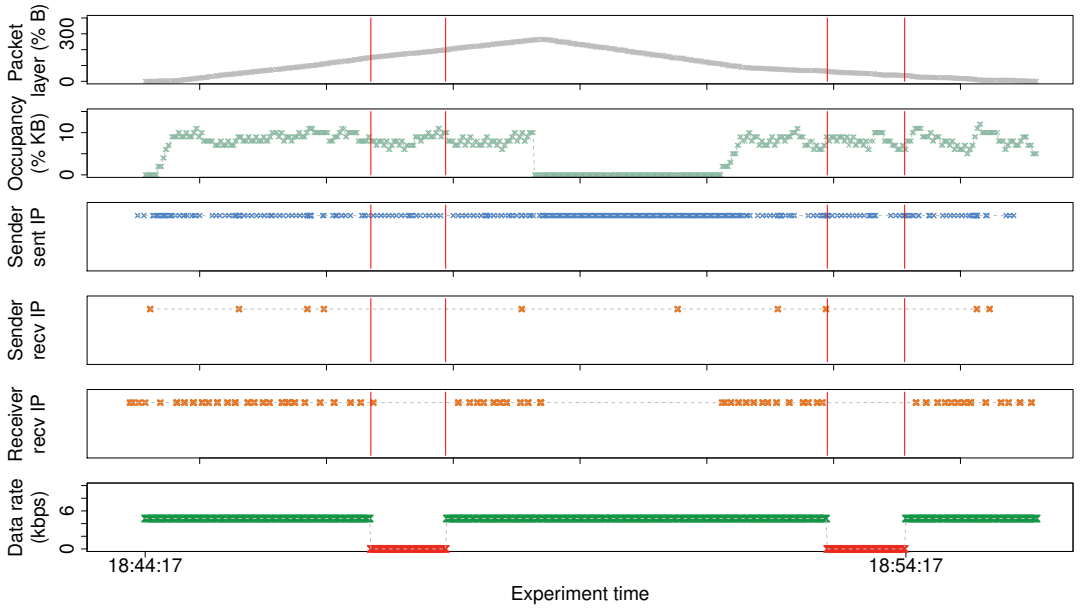

Figure 5.7 System behaviour for two disconnections

ii) packets intercepted at the sender, and iii) packets intercepted at the receiver at a given instance of time respectively. Red vertical lines across all layers help to visualize its behavior during two link disconnection intervals. The last row depicts the data rate in the network where the green plot indicates 4.8 kbps and red plots indicate disconnection in the link.

The system identifies the first disconnection after a time-lapse of 129 seconds (from the start time of the first disconnection) which is highlighted in the radio buffer occupancy layer, where the packets are dropped (occupancy = 0). The same behavior was expected for the second disruption. However, it was not observed (occupancy $\neq$ 0), indicating that the system either does not identify the link disconnection or identified it after a long time-lapse. In either case, the system suffers from packet loss. The deployment of unreliable UDP protocol does not mitigate the problem of packet loss which necessitates the requirement of a disconnection identification model. Some of the existing solutions to mitigate this problem are the implementation of a reliable transport protocol like TCP and deploying active network monitoring tools. The former creates additional delays for connection re-establishment while the latter adds overhead to the network, which makes both of them unsuitable for frequent disruption and bandwidth-constrained network as in TACTICS. Therefore, there is

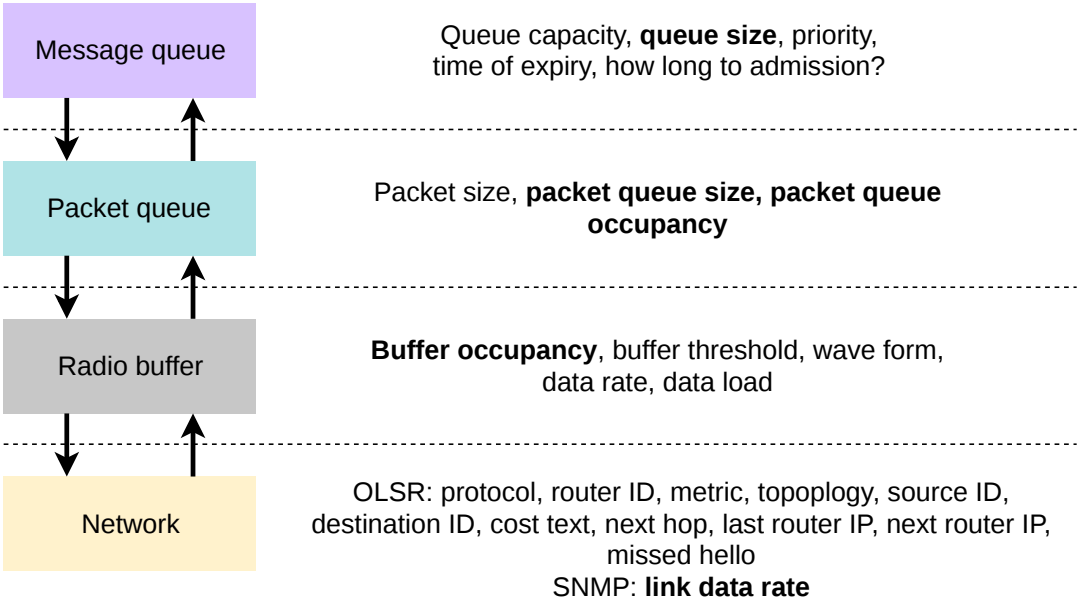

Figure 5.8 Features from different layers for learning model

a need for a model to identify disconnection based on the features retrieved within the system and help to mitigate its impact. The following observations were made from the experiment shown in Figure 5.7: i) the packet queue occupancy and the radio buffer features can be used to identify disconnection, ii) the IP packets sent at the sender is insensitive to link disconnection, iii) the IP packets are not intercepted during disconnection and iv) the receiver does not intercept any packet during a disruption. Based on the observations made, the disconnection can be identified by i) using the features from different layers as an input to the learning model, and ii) a time-based anomaly detection model can be used to identify disconnection. We discuss both of them in the following sections.

5.2.3 Machine learning approach

As seen in the initial experiment depicted in figure 5.7, the hierarchical queuing model features from different layers could be used to identify network disconnection. The idea of using the input features from different layers to classify if the link is connected or not is relatable to using binary classifiers. With this motivation, the next step was taken towards using suitable offline machine learning models to identify link disconnection based on the input features from the queuing model. Figure 5.8 shows the different layers of the data pipeline (left) starting from the message queue, packet queue, radio buffer, and network. The parameters retrieved from each of these layers are shown on the right.

These parameters can be treated as the input features of the machine learning model. Therefore, the feature extraction from the cross-layer information exchange constitutes the *Data acquisition* (step 1) as shown in figure 5.9. This is followed by the *Feature extraction and pre-processing* (step 2) wherein only the selected features for further processing are retained from the entire set of features, also known as feature reduction. As a part of the pre-processing, certain necessary steps are performed, for example, the time-stamp of all the features is maintained in a specific format,

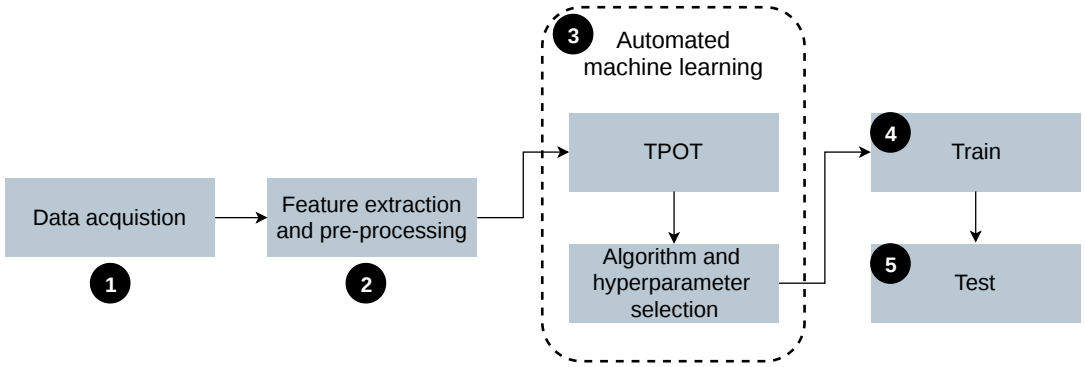

Figure 5.9 Machine learning model

calculating additional features such as average/slope within a time window. The next step is to select a suitable algorithm and corresponding hyperparameters (step 3) which is subsequently trained and tested (steps 4 and 5). Step 3 is explained in detail as follows. To get started with the hands-on experience of using the machine learning models and understanding the changes required in the data pre-processing steps, Automated Machine Learning (Auto-ML) was used. Auto-ML has gained importance in recent years as it provides the opportunity for using the models to solve the problem, without having prior expertise in machine learning. This process automates the selection of the algorithm and their hyperparameters that is suitable to attain higher performance metrics for the given input and target features. Therefore, the usage of Auto-ML approaches helped to define the possible useful algorithms and hyperparameters which outputs the best score and can be used to overcome the problem. In this sense, there are several Auto-ML tools available nowadays, but, in this study TPOT [14] tool was used. This tool is open-source deployed in Python language, which also has *scikit-learn* used for data prediction, classification, and analysis. The ease of use in Python and the availability of multiple data analysis tools makes TPOT a suitable choice for this study. The input features and expected target values are given to TPOT, which fits both the former and latter data to output the best-suited machine learning model for the given problem. This process was conducted several times with different input and target datasets to obtain the suggested models. Three models were suggested during this process which include *Gradient Boosting*, *Random Forest*, and *XGBoosting*. The pipelines of these models are implemented along with the suggested hyperparameters to validate the experiments.

5.2.4 Time-based anomaly detection

IP packets intercepted at the sender layer in figure 5.7 shows a time anomaly (not receive any packets) during disconnection. Keeping this as the motivation along with the consideration of not getting expected results from the machine learning approach, an alternate approach of time-interval anomaly detection based model was designed to identify the time interval of disconnection and predict lost packets. This mechanism is explained using a reference scenario consisting of n nodes where a sender node sends/receives packets to/from other nodes in the network. This is explained using Figure 5.10 which is divided into four columns where the first column represents the time series at the sender, the second shows the pack-

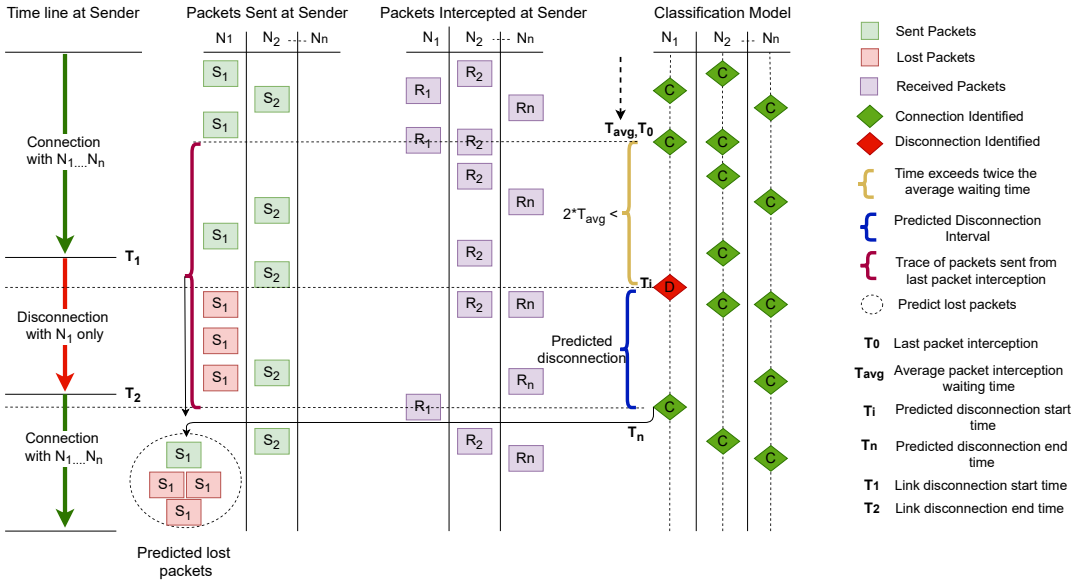

Figure 5.10 Disconnection Classification Model

ets sent at the sender and the third column shows the packets intercepted at the sender. Each of these packets is highlighted with numbers 1/2/n to differentiate the packets sent/received to/from the respective nodes. The packets sent/received could be an IP packet or a message sent for Blue Force Tracking (BFT)/Friendly Force Tracking (FFT) (routine messages exchanged between nodes) or route probing from the pro-active protocol. The last column shows the classification model which runs an instance for each of the connected nodes in the topology. For explaining the scenario, the classification model is used as a reference from the sender side and the link disconnection is highlighted between the sender and node N_1. Whenever this classification model intercepts a packet from a particular node, it calculates the time difference between the previous and the current intercepted packet. In addition, it keeps a track of the moving average of the time differences to calculate the average waiting time (threshold) for intercepting the next packet. In this example, the link disconnection is simulated from time interval T_1 to T_2. At time T_0, the sender intercepts a packet from the receiver (N_1) and calculates the corresponding average waiting time (T_{avg}) based on the previous packet interception intervals. If the sender does not intercept any packet from the receiver (N_1) within a time window of twice the average waiting time ($2 * T_{avg}$), then it identifies disconnection at the time (T_i). Note that (T_i) is identified after a delta time-lapse from T_1. The link is re-established at T_2 after which the sender intercepts a packet from the receiver (N_1) at time T_n. This marks the end of the predicted disconnection time from T_i to T_n. Note that T_n is identified after a delta time-lapse from T_2. Since a disconnection is predicted between the two packet interception intervals (T_0 to T_n), the trace of all the packets sent during this interval is predicted to be lost. This may include the packets that have been successfully transmitted. This model works well when there is active to-and-fro traffic between the sender and receiver and can converge to find the link re-establishment quicker than the routing protocol. However, this might not detect changes when the disconnection lasts for small intervals such as a few seconds of time.

6

Implementation

This Chapter discusses the implementation details of the design phase discussed in the previous Chapter 5. A control mechanism is implemented at the packet handler to regulate the dynamic shaping of the data flow based on the system and network parameters. To identify and mitigate the impact of disconnection, the implementation involves the deployment of two suitable models to identify disconnection. The former is accomplished by an offline binary classifier of the machine learning approach while the latter deploys a time-based anomaly detection model based on the packet interception.

6.1 Dynamic adaptation of dataflows

A packet handler service is implemented in TACTICS framework which intercepts the packets from the operating system kernel. Next, the intercepted packets need to be transmitted to the buffer by monitoring the network layer and buffer capacity. The implementation avoids data congestion by regulating the data flow even with the ever-changing network conditions. In order to accomplish this task, cross-layer information exchange is leveraged to improve the network and system awareness across the hierarchical queuing model. Using the cross-layer information exchange, the link data rate and IPI are computed for accomplishing the dynamic data flow. They are discussed in detail in the following sections.

6.1.1 Link data rate

Using the cross-layer information exchange, the buffer occupancy from the radio and data flow of the IP packets being transferred from the packet layer is retrieved. The link data rate Δd (kbps) is computed using Equation 6.1, where p_s is the total number of packets sent to the radio buffer, P is the packet size (B), ΔB is the buffer occupancy (%), B is the capacity of the buffer (in our test-bed 128KB) and

Δt is the time window (seconds). The parameters in Bytes are converted to bits for computing the link data rate in kbps.

$$\Delta d = \frac{(p_s * P) + (\Delta B * B)}{\Delta t} \tag{6.1}$$

This is implemented as an offline model, wherein the data rate is computed for a particular time window t. In TACTICS framework, the link data rate is retrieved from OLSR and SNMP protocol. However, in case of the absence of access to the protocol parameters, this method can be used to compute the data rate without having to consider any features from the receiver side. Also, this helps to visualize the network without any additional deployment of an active or passive tool for network monitoring. Moreover, these computations can be given as an input to cross-layer exchange to improve network awareness within the system to shape the data flow in the pipeline.

6.1.2 Inter-Packet Interval

Based on the analogy discussed in Section 5.1.2, to maintain the inflow of the data to the radio buffer, a control mechanism is implemented at packet handler based on: (i) the data rate in the network and (ii) buffer metrics: threshold and occupancy (radio). This analogy is formulated into mathematical equations as follows.

$$\alpha_r = \Delta t / \frac{(\Delta t * \Delta d) + ((b - \Delta B) * B)}{P} \tag{6.2a}$$

$$\text{IPI} = \begin{cases} \alpha_r + l & \text{if } b - \Delta B \ < b/3 \\ \alpha_r & \text{otherwise} \end{cases} \tag{6.2b}$$

The goal of the computation of IPI in Equation. 6.2b is to increase the accuracy of computing the link data rate thereby, shaping the data flow based on both network and system awareness. In Equation 6.2a, α_r is the delay in seconds where Δt is the time interval (sec), Δd is the network data rate (kbps), P is the packet size (bits), b is the buffer threshold, ΔB is the buffer occupancy and B is the capacity of the buffer (in our test-bed 128KB). The link data rate Δd is compiled by SNMP protocol and is accessed at the middleware using the contextual monitoring interface. However, if there is no access to the data rate in the network, then the link data rate is computed using Equation 6.1. Until 2/3 of the buffer is filled, IPI is comparatively smaller to fill up the buffer. Once the space left in the buffer (threshold - occupancy) is lesser than 1/3 of the threshold, additional latency of l (seconds) is introduced. This is shown in the Equation 6.2b with the goal to increase the precision of monitoring the buffer overflow. l should be an optimum value that can avoid buffer overflow.

Generally, the buffer metrics are retrieved every second. However, if for any reason the metrics could not be retrieved, then the precision of monitoring the buffer is hampered. This is crucial when the occupancy is very close to the threshold. For example, with the given threshold b at a time t_0, the occupancy is $b - 3$ whereas at time t_1, the occupancy is b, which means the transmission of packets to the buffer

can exceed its capacity. However, if the metrics could not be retrieved at time t_1, then the packet handler assumes that occupancy is still $b - 3$ and will transmit the packets. To avoid this problem, especially when the occupancy is higher than $\frac{2}{3}$ of the buffer threshold, latency is added. It is based on the assumption that even if a first reading (from radio) was missed, latency is added to verify the buffer metrics in the subsequent readings.

6.2 Disconnection identification

As discussed in section 5.2 of the previous chapter, there is a need for deploying a disconnection identification model without adding overhead in the network. In this section, the implementation of link disconnection simulation is discussed. It is followed by two models that can be used for link disconnection identification based on the features extracted from the system using the cross-layer information exchange. The link simulation and identification models are developed outside of the middleware to enable the re-use of these models in other architectures.

The simulation of the link disconnection is caused by using the hardware model. This model is described in section A.1 of Chapter A. The next step is to use the network states as in Figure 5.6 as the input to the statistical distributions to create sequences of network states. The network model in [32], uses Markov chain to produce multiple sequences of varying data rate states with static time interval transitions. This periodic time interval can be improved by introducing variability in the time interval between state transitions. Therefore, the model is improved to include variability in the state transition time to follow additional statistical distributions. These statistical distributions are developed using R script for generating both network state sequence and state transition time intervals. Although these statistical distributions might not match the fidelity of the real-time data, these distributions can still be used as an alternative approach to introduce variation in the network condition in case of no access to the on-field experimental data. Additionally, the distributions can be applied to other network parameters by researchers without requiring any additional installation.

This is explained using Algorithm 1. Steps 1 to 7 describes the procedure to use Gaussian Distribution. The input for this procedure is n which is the length of the network sequence, mu is the mean, $sigma$ is the standard deviation, $lower$ and $upper$ represents the lower limit, and the upper limit of the values in the network sequence. For example in our case, the set of network state number ranges from 0 to 5 (Figure 5.6), therefore, the lower limit would be 0 and the upper limit is 5. $rnorm$ is the total number of samples to be generated. This is required to select a sequence from a set of sample sequences that match the requirement of the upper and lower limits. In step 2, $rnrom$ number of Gaussian Distribution sequences is generated with length ($norm$), mean (mu) and standard deviation ($sigma$). The sequence which satisfies the condition of the upper and lower limits are retained in step 3. The selected sequence is returned in step 5. Similar to the Gaussian Distribution procedure, another procedure for Binomial Distribution is described from steps 9 to 17. This procedure takes input parameters n which is the number of observations, $size$ is the number of trials, $prob$ is the probability of success, and $state$ is the

Input: Distribution parameters, length of sequence
Output: Distribution sequence

```
1: procedure GaussianDistribution(n, mu, sigma, lower, upper, norm)
2:     sequence = (rnorm(norm,mu,sigma))
3:     sequence = sequence[sequence <= lower and sequence >= upper]
4:     if  length(sequence) >= n then
5:         Return sample(sequence,n)
6:     end if
7: end procedure
8:
9: procedure BinomialDistribution(n, size, prob, state)
10:    sequence = (rbinom(n,size,prob))
11:    for i in 1:length(sequence) do
12:        if sequence[i] == 1 then
13:            sequence[i] == state
14:        end if
15:    end for
16:    Return sequence
17: end procedure
18:
19: networkIntervals = GaussianDistribution(n,mu,sigma,lower,upper,norm)
20: networkStates = BinomialDistribution(n,size,prob,state)
```

Algorithm 1 Statistical distributions for generating network states

data rate state number from 1 - 5. A binomial distribution sequence of length n is generated with probability *prob* for the occurrence of the disconnection (state 0) in step 10. Since the sequence consists of either 0 or 1 corresponding to disconnected or connected state respectively, the connected state is replaced with the input *state* number (from 1 to 5 data rate states) in steps 11 to 15. The generated sequence is returned in step 16. The network interval Gaussian Distribution and the network state Binomial distribution are generated in steps 21 and 22 respectively.

6.2.1 Machine learning approach

The implementation of the machine learning model is classified into four steps of *data acquisition*, *feature extraction and pre-processing*, *train the model*, and lastly to *test* the model as seen in Figure 5.9. As a part of the *data acquisition phase*, the features from the hierarchical queuing model and SNMP are acquired from the cross-layer information exchange as shown in Table 6.1. The first column highlights the layer whereas the second column shows the features extracted from the corresponding layers. The third column provides a brief description of the extracted features.

Since all these features are extracted from different layers, they need to be grouped together at a given instance of time. Therefore, the timestamp available at each layer are used to join the features to form the feature data set. The timestamp across all the nodes of the network are synchronized by deploying the Network Time Protocol (NTP) protocol. Moving ahead, certain features such as the timestamp and interface

Layer	Feature	Description	Selection
Message	Time stamp	Time at which message queue details were obtained	
Message	Interface	Network interface	
Message	Queue capacity	Maximum number of messages that can be occupied in the queue	✓
Message	Queue size	Current number of messages in the message queue	✓
Message	Priority	Priority of the message	
Packet	Time stamp	Time at which the packet layer details were obtained	
Packet	Interface	Network Interface	
Packet	Size	Packet queue size (B)	✓
Packet	Usage	Packet queue occupancy (B)	✓
Radio	Time stamp	Time at which radio buffer details were obtained	
Radio	Data throughput	SNMP link data rate	✓
Radio	Pending capacity	Pending out bytes in radio buffer (B)	✓
Radio	Queue occupancy	Queue occupancy in (%B)	✓
Radio	Load	Average of the data load	✓
Packet	Slope	Slope of the change in packet size	✓
Radio	Slope	Slope of the change in radio buffer occupancy	✓

Table 6.1 Features extracted from different layers for learning model

do not help in training the model as both of them are not influenced by the network conditions. Therefore, the dataset is reduced by selecting only those features that are required for further analysis, and this is highlighted with ✓ in the fourth column of Table 6.1. Next, only the required rows are retained. For example, the rows of the feature data set before and after the experiment are removed in pre-processing. In addition, the slope of the packet size and the radio buffer occupancy is calculated (last two rows from bottom in Table 6.1). This is done to distinguish between the ascent and descent of the slope based on the underlying network conditions. It is explained in Figure 6.1 which shows the behavior of the packet queue and radio buffer for the varying data rates in the network. The first layer (from below) shows the data rate (kbps) in the network whereas the second layer shows the radio buffer occupancy (%KB) and the third layer indicates the packet queue occupancy (%B). Across all the three layers, the color of the plots is based on the model described in Figure 5.6. The packet layer and the buffer occupancy plots are highlighted with two regions, 1 and 2 to show the difference in the behavior of the curve based on data rates in the network. The descent of the slope of the packet queue for region 1 is smaller when compared to the slope in region 2. This is because the former consists of a lower range of data rates ($<= 2.4$ kbps) while the latter region consists of a higher range of data rates ($>= 4.8$ kbps). For this reason, the range of buffer occupancy is between 8 to 13 for region 1 whereas for region 2 the range of buffer occupancy is 6 to 10. This re-iterates the analogy (used for link data rate computation) that the number of packets transmitted by the buffer is higher with higher data rates. These features are selected as the input features of the learning model.

A logical question arises here that the data rate is given as one of the inputs to the learning model. Ideally one can expect data rate to be 0 during disconnection making it redundant to be using the machine learning model. However, in TACTICS framework, the system does not realize network data rate change. For example, suppose at time t_0, the data rate in the link is 4.8 kbps and disconnection is simulated from time t_1 to t_2 . During the disruption interval, the compiled link data rate is still 4.8 kbps. Going ahead, three machine learning models were implemented based on the initial experiments conducted using Auto-Machine Learning (Auto-ML) tool TPOT.

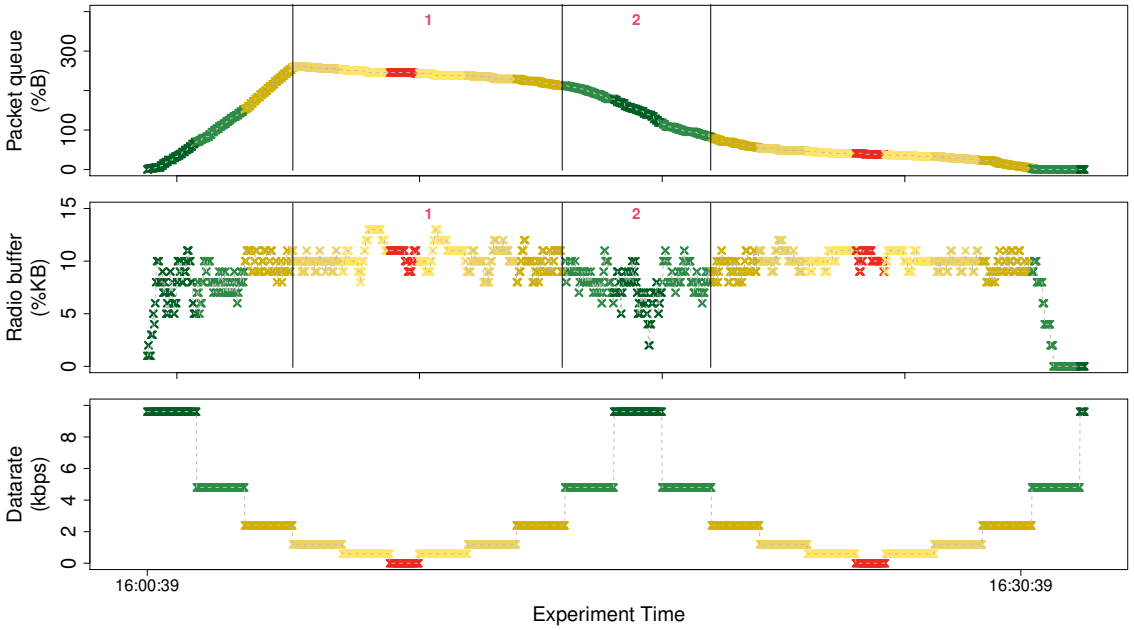

Figure 6.1 Slope of the curves for varying data rates

Input: Training features, training target, testing features, testing target
Output: Predicted target, classification report

1: exportedPipeline = make_pipeline(*Classifier (hyperparameters)*)
2: exportedPipeline.fit(trainingFeatures, trainingTarget)
3: predictedTarget = exportedPipeline.predict(testingFeatures)
4: print(classificationReport(testingTarget,predictedTarget))

Algorithm 2 Machine learning classifier implementation

The dataset sample is categorized into training features, training target, testing features, and testing target. The training features and training target are given as input to the classifier to train the model. When the trained model is given with testing features as input, the prediction labels are given as output in the predicted target. The analysis is done based on the comparison between the predicted and testing/actual target. This process is explained using the pseudo-code in Algorithm 2. In step 1 of the algorithm, the pipeline of the learning model classifier is specified with the hyperparameters. In this study three classifiers are used i) Gradient Boosting, ii) Random Forest and iii) XGBoosting (KNN and XGB Classifier). The hyperparameters used for the respective models are highlighted in Table 6.2. The first column of this table specifies the classifier used, the second column corresponds to the hyperparameter and their values are tabulated in the third column. After creating the pipeline in step1, in step 2 of Algorithm 2, the training and testing features are given as input to train the classifiers. After the model is trained, the testing features are given as the input to get the predicted target in step 3. The classification report is printed in step 4 to compare the predicted target with the actual target in step 5.

Classifier	Hyperparameter	Value
Gradient Boosting	Learning rate	0.1
	Maximum depth	8
	Maximum features	0.45
	Minimum samples leaf	6
	Minimum samples split	18
	n Estimators	100
	subsample	0.8
Random forest	Bootstrap	False
	Criterion	Entropy
	Maximum features	0.70
	Minimum samples leaf	1
	Minimum samples split	12
	n Estimators	100
KNeighborsClassifier	n Neighbors	12
	p	1
	weights	distance
XGBClassifier	Learning rate	0.001
	Maximum depth	9
	Minimum child weight	1
	n Estimators	100
	n Thread	1
	Subsample	0.75

Table 6.2 Learning model hyper parameters

6.2.2 Time-based anomaly detection

This section describes the implementation of a stand-alone python script to identify link disconnection and re-transmit the lost packets. This is developed based on the design of the model in section 5.2.4 wherein all the nodes of the network run an instance of this script for all other connected nodes in the network. It is explained using Algorithm 3 with only 2 nodes $Node1$ and $Node2$ with the script running on $Node1$ for detecting a link disconnection with $Node2$ (for better understanding). $Node1$ and $Node2$ are transmitting packets between each other. The algorithm is explained from the perspective of $Node1$ where a *"sent"* packet refers to the packet sent from $Node1$ to $Node2$ and a *"received"* packet refers to the packet received at $Node1$ from $Node2$. The frequency and the time intervals between the packets received at $Node1$ from $Node2$ help to identify if the link is connected/disconnected.

Using the Pyshark module, all the packets (*packet*) in the network are continuously sniffed in step 1. If a packet was sent from $Node1$ to $Node2$ (step 4), the packet details such as the $packetNumber$, $frameNumber$, $sourceNumber$, $destinationNumber$, and $rawPayload$ are stored in a table (steps 5 - 9). The timestamp of the current sent packet ($currSentTime$) is noted in step 10. If the difference of the time intervals between the sent packet ($currSentTime$) and the $prevRecvTime$ (last received packet from $Node2$) exceeds twice the $prevRecvAvg$ (averages of the time difference of the received packets from $Node2$) then the predicted disconnection start time is printed in step 12. If a packet was received from $Node2$ (step 14), then timestamp of this packet interception $currRecvTime$ is noted in step 15. The difference in the time interval of the current received packet and the previously received packet ($diffRecvTime$) is calculated in seconds in step 16. If this value exceeds twice the

average of time differences of the received packets from $Node2$ ($2 * prevRecvAvg$) as in step 17, it indicates that a previously identified link disconnection (step 11) has ended with successful packet interception from $Node_2$ (end of predicted disconnection time) (step 18). Hence, the trace of packets sent from $Node1$ to $Node2$ between the previous and current timestamp of the received packets from $Node2$ are written to the file in steps 20 - 22.

After the packet details are successfully written to the file, the contents of the packet table are cleared in step 24. Steps 26 - 36 are done to calculate the moving average of the time differences of received packets. The time differences between the received packets are maintained in a list ($listdiffRecvTime$) of length n. Here, n represents the total number of previously received packets taken into consideration for calculating the average time ($prevRecvAvg$). If any of the values of the list ranging from 1 to n is missing (step 26), then the corresponding index value ($listdiffRecbTime$) and ($prevRecvAvg$) are updated in steps 27 and 28. If the values at all indices of the list are already filled, then the current values at all indices of the list are updated by the value of their next index (step 31) whereas the value of the list at the last index is updated by the current time difference (step 32). This is done to update the values of the indices with the sliding window approach. The sum of the values at all the indices ranging from 1 to n are added ($sumdiffRecvTime$) in step 33 and their corresponding average value is calculated in step 35. The previously received packet time is updated with the current received packet time in step 37. The output of this approach predicts disconnection start and end time intervals along with the list of lost packet details in a text file.

Input: Packet interception time, packet details
Output: Predicted disconnection time, lost packet details

```
1: cap = pyshark.LiveCapture(interface)
2: while True do
3:     for  packet in cap.sniffContinuously(): do
4:         if packet.srcAddress = Node1 and packet.dstAddress = Node2 then
5:             packet.table.append(packetNumber)
6:             packet.table.append(frameNumber)
7:             packet.table.append(sourceNumber)
8:             packet.table.append(destinationNumber)
9:             packet.table.append(rawPayload)
10:            currSentTime = packet.sniffTimestamp
11:            if (currSentTime - prevRecvTime) > 2*(prevRecvAvg) then
12:                Print(Predicted disconnection start time, currSentTime)
13:            end if
14:        else if packet.srcAddress = Node2 then
15:            currRecvTime = packet.sniffTimestamp
16:            diffRecvTime = (currRecvTime - prevRecvTime).totalSeconds()
17:            if diffRecvTime > 2*(prevRecvAvg) then
18:                Print(Predicted disconnection end time,currRecvTime)
19:                for row in range(len(packet.table) do
20:                    for col in range(len(packet.table.row)) do
21:                        filehandle.write(packet.table.row.col)
22:                    end for
23:                end for
24:                packet.table.clear()
25:            end if
26:            if Any value of listdiffRecvTime[i] from 1 to n == 0 then
27:                listdiffRecvTime[i] = diffRecvTime
28:                prevRecvAvg = diffRecvTime/i
29:            else
30:                for i in 1:n do
31:                    listdiffRecvTime[i] = listdiffRecvTime[i+1]
32:                    listdiffRecvTime[n] = diffRecvTime
33:                    sumdiffRecvTime = sumdiffRecvTime + listdiffRecvTime[i]
34:                end for
35:                prevRecvAvg = sumdiffRecvTime/n
36:            end if
37:            prevRecvTime = currRecvTime
38:        end if
39:    end for
40: end while
```

Algorithm 3 Identify disconnection and re-transmit lost packets

7

Evaluation

The network identification and adaptation mechanisms discussed in the previous chapter are evaluated and the experimental results are discussed in this chapter. The robustness of the tactical system is tested with two scenarios of varying data rates and link disconnections. In the former, the proposed dynamic adaptation mechanism of the data flow is compared to the Threshold Shaping (baseline approach) [33] with constant data rate conditions. The experiments are further extended to observe the sensitivity of the adaptation mechanism to ever-changing data rates in the network. For identification of link disconnection, the machine learning approach and time-based anomaly detection model is evaluated by simulating link disconnection. The experimental results indicate that the latter is a suitable approach and its corresponding predictions can be used by the system to increase the robustness.

7.1 Experiments with varying link data rates

This section discusses the adaptation mechanism of the system to varying data rates in the network. It is discussed in three steps i) validate the observation capacity of the system to compute the network metrics without adding additional overhead in the network, ii) quantitative comparison of IPI with the baseline approach to shape the data flow and iii) test sensitivity of IPI to adjust the flow of the data dynamically to network variation.

7.1.1 Data rate computation

The data rate is retrieved from the radio using SNMP protocol and can be accessed using the cross-layer information exchange via contextual monitoring interface as described before in TACTICS architecture. In case of absence of such an interface with the network, an alternative approach is validated using experimental results

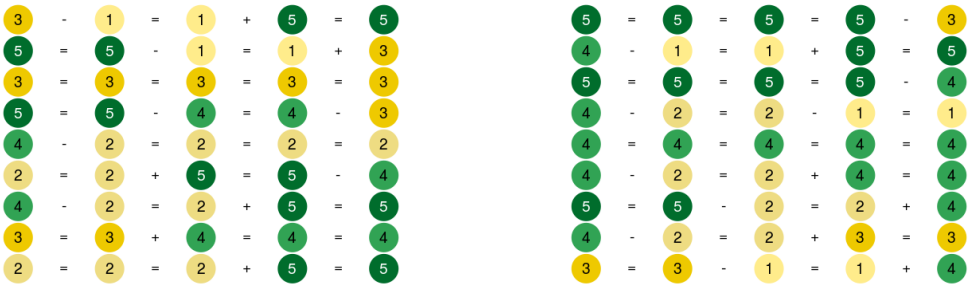

Figure 7.1 DR_1 (left) and DR_2 (right) data rate patterns

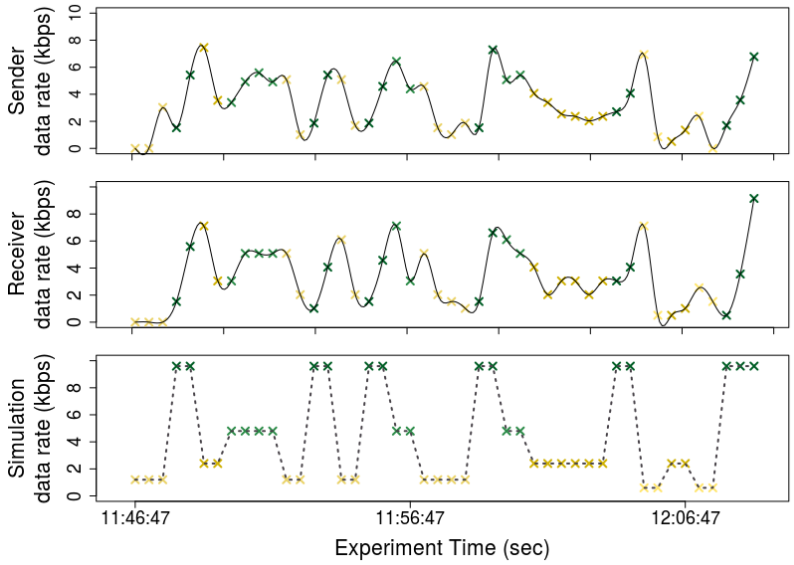

Figure 7.2 Data rate computation for DR_1 (Experiment 1)

that suggest that the system can compute the data rate based on the buffer metrics and the data flow of the packets at the sender.

In this sense, two experiments are conducted by simulating random data rate sequences in the network, and the sensitivity of the system to detect the data rate changes are evaluated. Two data rate patterns used in the experiments are shown on the left and right of Figure 7.1. Each of these figures consists of 45 states where each of the states corresponds to the data rate shown in Figure 4.3. The sequence traverses from bottom to top across every row from left to right. The signs between the states indicate increase (+), decrease (-) or the same state transition (=). These data rate sequences are simulated in the network as a pre-requisite to test the sensitivity of the model to compute the link data rate.

Two experiments (Experiment 1 and 2) are conducted to validate the data rate estimation model by simulating the link data rate to vary in the network using DR_1 and DR_2 respectively. The result of the experiments are shown using Figures 7.2 and 7.3. These figures are divided into three rows with data rates in *kbps* plotted against time in *seconds*. The first row from below shows the simulated data rate in the network with data rates varying from 0.6 to 9.6 kbps in a defined pattern (DR_1 and DR_2). The second row (from below) shows the data rate computed at the receiver side for validation, whereas the third row depicts data rate calculated at the sender

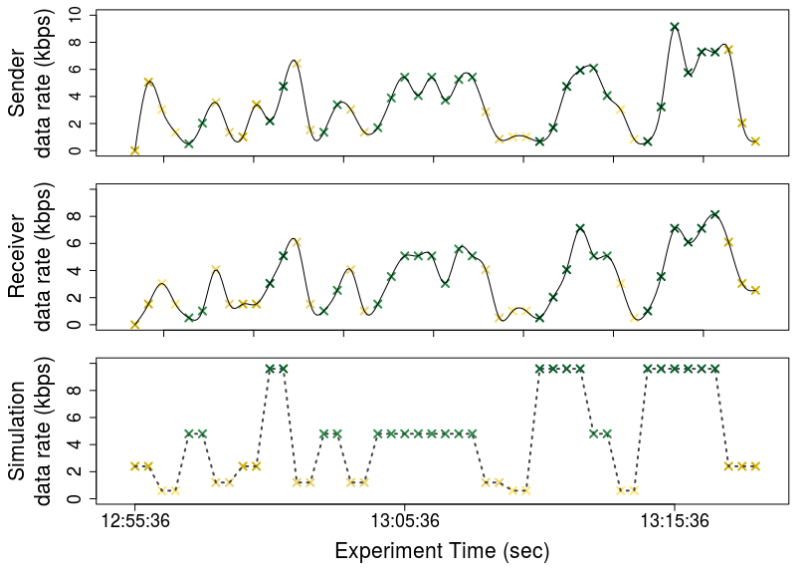

Figure 7.3 Data rate computation for DR_2 (Experiment 2)

based on the packet flow and the buffer occupancy computed using Equation 6.1. The data rate at the sender is calculated with a moving time window of 30 seconds. The simulated data rate is nominal which means it is the maximum available data rate in the network, however, this can be affected by interference, latency, and noise in the network. Therefore, the actual link data rate can be calculated based on the packets intercepted at the receiver. Hence, the nominal threshold is taken for reference to show the upper limit of the bandwidth supported in the network, but, the estimated data rate at the sender is validated against the computed data rate at the receiver.

The estimated data rate pattern at the sender is similar to the data rate pattern of the receiver however, there are minor variations that are calculated as the deviation and tabulated in Table 7.1. This table consists of the three columns for both experiments with DR_1 and DR_2 with the rate at the sender, receiver, and deviation respectively. The average deviation is calculated after every five data rate computation. The deviation never crosses more than 1 kbps, indicating that the data rate computation using the buffer metrics can be used by the middleware for visualizing the network metrics. This approach is dependent on the cross-layer information exchange with the interface to the radio, but, it does not add additional overhead in the network. This can be used for shaping the data flow and can be given as input to the routing protocol estimation.

7.1.2 Dynamic adaptation of the data flow with IPI

The first experiment is conducted to compare Threshold Shaping with IPI. In this experiment a constant data rate of 4.8 kbps is maintained in the network and the threshold is set to 10%. A 500KB message load is transmitted through the pipeline that can stress the store-and-forward mechanism. The results of the experiment are shown in Figure 7.4 with buffer occupancy (%KB) plotted over the experiment time (seconds). Threshold Shaping indicated by red plots takes about 15 minutes

Experiment 1 with DR_1			Experiment 2 with DR_2		
$Rate_S(kbps)$	$Rate_R kbps)$	$Dev(kbps)$	$Rate_S(kbps)$	$Rate_R(kbps)$	$Dev(kbps)$
1.99	1.42	0.63	1.99	1.32	0.74
4.98	4.67	0.37	2.27	1.93	0.81
3.65	3.45	0.67	3.25	3.35	0.37
3.93	4.26	0.47	2.67	2.54	0.54
2.67	2.54	0.74	4.77	4.77	0.54
4.67	4.67	0.40	2.23	2.33	0.37
2.53	2.64	0.64	3.82	3.76	0.67
3.01	3.04	0.17	2.37	2.64	0.40
1.79	1.82	0.64	7.38	6.91	0.95
$MeanDev(kbps)$		0.53			0.60

Table 7.1 Data rate estimation comparison of Experiment$_1$ and Experiment$_2$

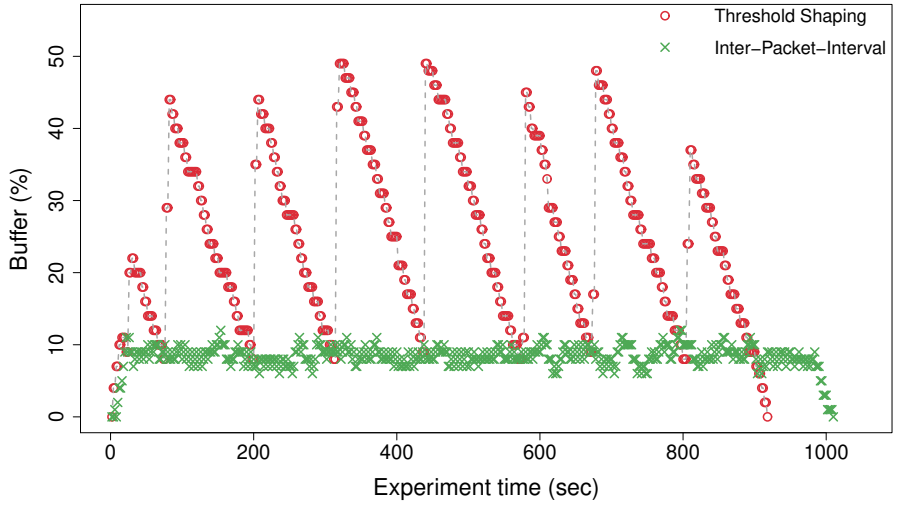

Figure 7.4 Comparison of Threshold Shaping with IPI

to complete the experiment. On the other hand, shaping with IPI indicated with green plots takes about 16 minutes to complete the experiment. The Threshold Shaping shows a zig-zag pattern with peaks of buffer occupancy until 50% even though the threshold is set to 10%. On the other hand, the shaping with IPI indicates a consistent line with buffer occupancy around 10% with minor deviations. This indicates that data flow is sensitive to adhering to the threshold which makes IPI a better choice than Threshold Shaping. An important point to observe here is that, if the threshold was set to 60% and above (until 90%) the Threshold Shaping will suffer from packet loss in which case the peaks of the zig-zag pattern are expected to go beyond 100% leading to packet loss. On the contrary, even at 90% threshold, IPI maintains the buffer occupancy around the threshold mitigating the problem of packet loss.

In the next step, IPI is evaluated with the ever-changing network conditions. For this the link data rate is simulated to follow the sequences of D_2 and D_3 shown in Figure 4.4 and Figure 4.5 respectively. D_2 and D_3 are referred to as DR_3 and DR_4 in this chapter to follow uniformity while explaining the results. DR_3 is used for the first experiment while DR_4 is used for the second experiment. A 500 KB message load and a threshold of 50% are used in both the experiments. To quantify the results, the following terminologies are used. The total time taken by the experiment is referred to as T_T (min). The time taken by the system to fill the buffer until threshold value

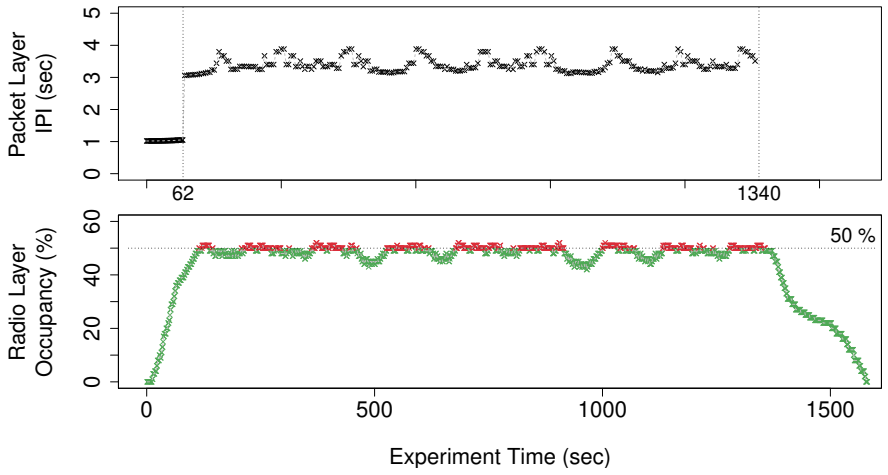

Figure 7.5 IPI with DR_3 data rate pattern

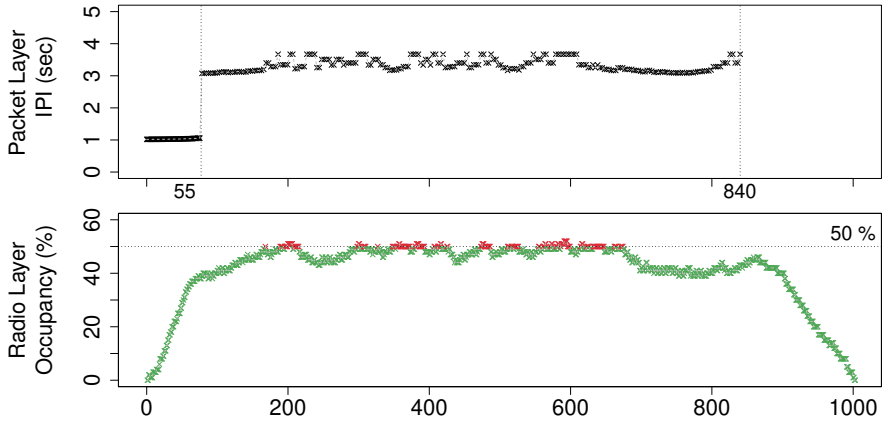

Figure 7.6 IPI with DR_4 data rate pattern

for the first time during the experiment is referred to as T_t (min). Larger value of T_t signifies that for a long time the system has not taken advantage of the allowed capacity of the buffer. The experiment with DR_3 was completed in $\approx$ 26.3 min whereas DR_4 takes $\approx$ 16.7 min because DR_4 has a higher average data rate than DR_3. Figure 4.4 and Figure 4.5 consist of two plots. The first plot (from below) plots the radio buffer occupancy (%) over time whereas the second plot indicates the computed Inter-Packet-Interval in seconds. The buffer occupancy is highlighted in green or red plot to indicate if the buffer occupancy is below or above the threshold respectively. It is observed that T_t for DR_3 and DR_4 are $\approx$ 1.9 min and $\approx$ 2.8 min respectively. This indicates that within a short interval of time, the buffer gets filled up leading to efficient usage of the buffer capacity. As seen in Figure 7.5, until the 62^{nd} second the computed IPI is approximately 1 sec until $2/3^{rd}$ of the buffer is filled (approximately 33%). After this, an additional latency of 2 seconds is added because, once the buffer gets closer to the threshold, the control mechanism needs to be more cautious to avoid overflow.

Similar observations can be made in the experiment with DR_4 pattern. In Figure 7.6, until the 55^{th} second the computed IPI is approximately 1 sec until $2/3^{rd}$ of the buffer

Figure 7.7 DR_5 data rate sequence (left) and TR_5 time sequence (right) for link disconnection

is filled (approximately 33%). Additionally, it is identified that IPI is sensitive to network changes by visualizing small variations of the crests and troughs in IPI of Figure 7.5. In short, the experimental study indicates that until the buffer threshold is reached, priority is given to buffer metrics (to fill the buffer close to the threshold), thereby the IPI is smaller (smaller T_t). Once the occupancy reaches closer to the threshold, then IPI is dependent on both data rate and threshold. The experimental study and comparison support the hypothesis that cross-layer IPI with features from different layers proves to be more robust to ever-changing scenarios rather than the Threshold Shaping.

7.2 Experiments with link disconnection

In this section, the robustness of the system to network disconnection is validated. This is carried out in three steps as i) simulating link disconnection with the hardware model using known statistical distributions, ii) validate the precision of the machine learning approach to classify disconnection and iii) test the sensitivity of the time-based anomaly detection model along with the precision of predicting the lost packets.

7.2.1 Network disconnection

In this case study, the ever-changing network conditions comprise of network states ranging from 0 to 5 (varying data rates (1-5) and disconnection (0)). Since the robustness of the system to dynamically regulate the flow of the data based on the network data rate is already validated in the previous section, the next experiment was conducted with a simplified test case for identifying the link disconnection. This case consists of only two valid states: connected and disconnected. The former signifies a constant data rate while the latter represents link disruption. A sequence consisting of only two states represents a Binomial Distribution wherein 1 represents the connected state (data rate state) and 0 indicates link disconnection. Therefore, the implementation discussed in Algorithm 1 is used to generate a sequence of network states using Binomial Distribution whose state transition intervals are generated by Gaussian Distribution. Both of them are represented on

Phase	Total sample	Label: Connection (0)	Label: Disconnection (1)
Training	5000	4410	590
Testing	1004	885	119

Table 7.2 Learning model statistics

the left and right of Figure 7.7 respectively. Both sequences consist of 100 states starting from bottom to top traversing every row from left to right. The network sequence DR_5 (left) represents states with the occurrence of 5 link disconnections (0:red) in between constant data rate (4:green) of 4.8 kbps, whereas the network interval sequence TR_5 (right) indicates the corresponding time interval transitions in seconds. For example, the first state (4) is simulated for 38 seconds whereas the fourth state corresponding to disconnection (0) is simulated for 100 seconds. For better visualization, all the disconnected states are matched with the same color in both the sequences (red).

7.2.2 Learning approach

The objective of developing the machine learning model is to build an offline classification approach to detect link disconnection based on the input features from different layers of the queuing model. The input features are taken from the different layers of the queuing model as specified in the *Selection* column of Table 6.1. These features are pre-processed to train three learning models: *Gradient Boosting*, *Random Forest* and *XGBoosting*. These models are used for binary classification of identifying if the link is connected or disconnected. Six different experiments were conducted following the distributions mentioned in the previous section. Each of them took approximately 30 minutes and five experiments are grouped into one training dataset (features and target) and one experiment as the testing dataset (features and target). The input consists of ten features as shown in Table 6.1 whereas the target comprises a single column of the binary classification labels. Label 0 is assigned for *connection* whereas 1 is assigned for *disconnection*. The evaluation is done in two steps starting with training the model followed by the testing process. Table 7.2 shows that the training is conducted on 5000 samples with 4410 and 550 classified into connection and disconnection respectively. The testing is done on 1003 samples where 885 are classified as connected whereas 119 are classified as disconnected.

The learning approach is evaluated with the three models, considering the metrics of *Precision*, *Recall*, and *F1-Score*. These metrics for the three models are shown in Table 7.3. The evaluation of these models is discussed as follows. The *Gradient Boosting* model has higher precision (88%), accuracy(96%) and F1-scores(92%) for classifying connection (Label 0). Similarly, *Random Forest* and *XGBoosting* show higher values of metrics for classifying connection. This is contradicted by the results for the classification of disconnection (Label 1). *Gradient Boosting* shows lower precision, accuracy, and F1-scores of 3%, 1%, and 1% respectively. Likewise, *Random Forest* and *XGBoosting* exhibit lower values of these metrics.

Based on the results, the current version of the classification model does not provide sufficient arguments to convince us to deeply explore this approach before reflecting on our naive solution and the potential areas to improve it are highlighted as

Model	Label	Precision (%)	Accuracy (%)	F1-Score (%)
Gradient Boosting	0	88	96	92
	1	03	01	01
Random Forest	0	88	94	91
	1	10	05	07
XGBoosting	0	90	87	89
	1	03	04	04

Table 7.3 Learning model metrics

follows. The training process might require more data samples. It also implies that training might require more input features that favor in better decision making of the classification or the current version of the input features might require additional data pre-processing and transformation (different dimensions). Moreover, the hyper parameters of these models can be varied and different classifiers can be deployed. Given the confined input feature and a wide range of possibilities of improving this approach within the given time frame, further analysis on machine learning was discontinued. However, this issue is left open as this requires further investigation and could still be an open research question for addressing in the future.

7.2.3 Time-based anomaly detection model

The experiment was conducted to validate the time-based anomaly detection model for link disconnection. The disconnection was simulated using the hardware prototype for a specified interval of time. The data rate sequence of 100 states follows a Binomial distribution as seen in Figure 7.7 (left) with connected (4.8 kbps) and disconnected states. The state transition intervals follow a Gaussian distribution as seen in Figure 7.7 (right). The experiment was conducted with two nodes ($Node_1$ and $Node_2$) connected by a single hop with bi-directional data traffic between the nodes. The detection model is run continuously on $Node_1$ throughout the experiment to intercept the packets sent/received to/from $Node_2$. The output of this model is the predicted start and end time of the disconnection along with the text file consisting of the packet details of the predicted lost packets. The results of the experiment are used for observation and analysis as follows.

Figure 7.8, Figure 7.9, and Figure 7.10 plots the observation of the link, packets sent, and received at $Node_1$ and prediction model as a time series of the experiment time. The three hour-long experiment is divided into three parts for analyzing the results. Figure 7.8 is considered for understanding the notations for the three sequences of figures. This figure captures two disconnections within the first time frame of the experiment. The first layer (from below) shows the link simulation layer (1) which distinguishes the link as connected (green) or disconnected (red) states based on the disconnection simulation prototype. The start and end of disconnection is highlighted with the notation of T_{1j} and T_{2j} respectively, where j indicates the disconnection number. The second layer (2) plots a mark (blue) if a packet was sent from $Node_1$ to $Node_2$ at a particular time instance. In addition, notation T_{3j} and T_{4j} are used to indicate the system's interpretation/realization of link disconnection which is done by monitoring the IP packets sent from $Node_1$. The third layer (3) plots a mark (orange) if a packet was received at $Node_1$ from $Node_2$. In this layer, the notations T_{0j} means the last (before disconnection) timestamp of the received

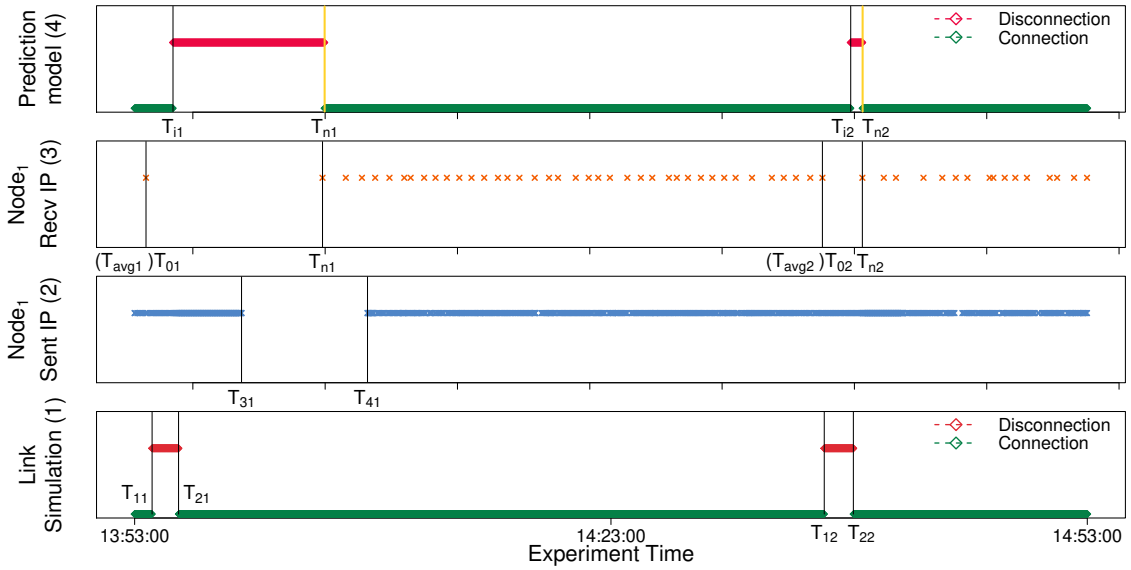

Figure 7.8 Disconnection identification experiment (part 1)

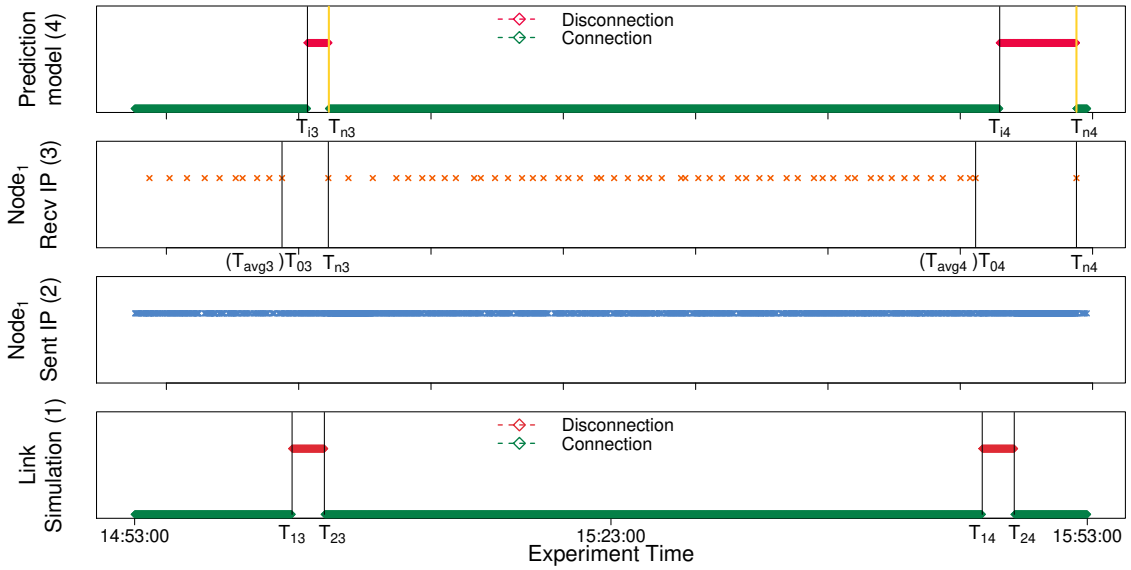

Figure 7.9 Disconnection identification experiment (part 2)

packet, T_{avgj} is the average of the difference in time intervals of received packets calculated at T_{0j}, and T_{nj} indicates the timestamp of the first packet received after disconnection. The fourth layer (4) shows the prediction of the link disconnection identification model whose plots are classified into connected (green) and disconnected (red) states. Notations T_{ij} and T_{nj} in this layer indicate the start and end of the predicted disconnection intervals. Note that $T_{ij} > 2 * T_{avgj}$ and T_{nj} is the same as in layer 3. T_{nj} at layer 4 is highlighted (yellow) to indicate that at this timestamp the model will write the list of traces of predicted lost packets into a file.

For each of the notations explained above, the corresponding timestamps are summarized in Table 7.4 for all the 5 disconnections. All these values are used to analyze the results shown in Table 7.5. Table 7.5 is used for comparing the advantage of deploy-

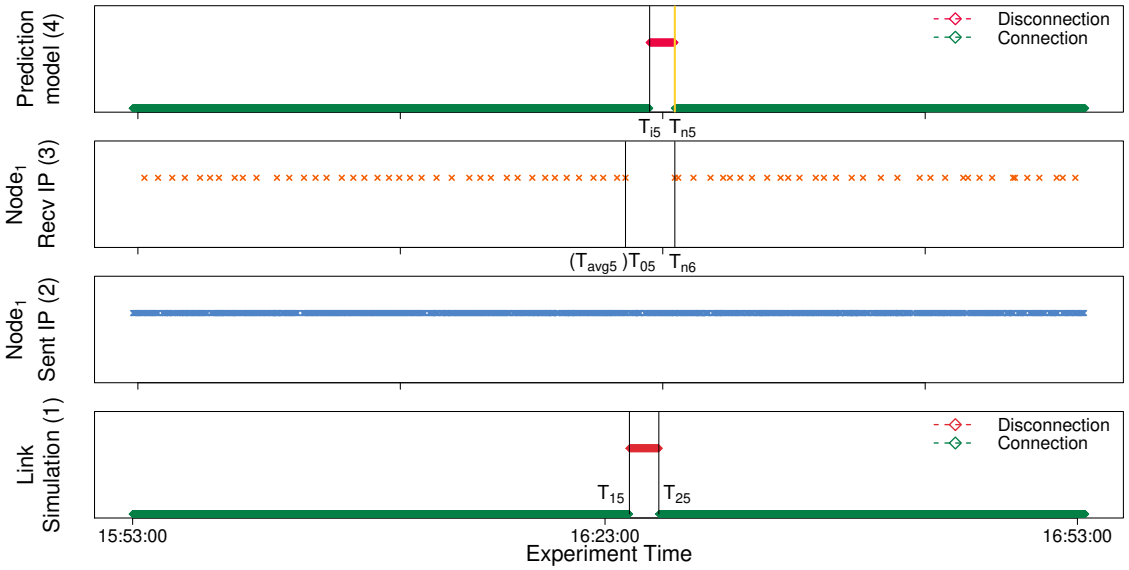

Figure 7.10 Disconnection identification experiment (part 3)

$Dis.Num.(j)$	T_{1j}	T_{2j}	T_{0j}	T_{avgj}	T_{ij}	T_{nj}	T_{3j}	T_{4j}
1	13 : 54 : 06	13 : 55 : 46	13 : 53 : 43	47.8s	13 : 55 : 25	14 : 04 : 58	13 : 59 : 44	14 : 07 : 40
2	14 : 36 : 26	14 : 38 : 16	14 : 36 : 19	53.2s	14 : 38 : 06	14 : 38 : 51	-	-
3	15 : 02 : 55	15 : 04 : 57	15 : 02 : 17	47.2s	15 : 03 : 53	15 : 05 : 14	-	-
4	15 : 46 : 23	15 : 48 : 24	15 : 45 : 58	44.8s	15 : 47 : 29	15 : 52 : 19	-	-
5	16 : 24 : 33	16 : 26 : 25	16 : 24 : 18	44.8s	16 : 25 : 50	16 : 27 : 26	-	-

Table 7.4 Time distribution of different notations used in link disconnection identificatio

$Dis.Num.(j)$	Simulation interval (s) $(T_{2j} - T_{1j})$	Time lapse in system predicted start time (s) $(T_{3j} - T_{1j})$	Time lapse in system predicted end time (s) $(T_{4j} - T_{2j})$	Time lapse in predicted start time (s) $(T_{ij} - T_{1j})$	Time lapse in predicted end time (s) $(T_{nj} - T_{2j})$
1	100	338	476	79	314
2	110	–	–	100	35
3	112	–	–	-58	17
4	121	–	–	66	230
5	112	–	–	77	61

Table 7.5 Comparison of sensitivity of system and prediction model for link disconnection

ing the prediction model to identify link disconnection. Totally five disconnections were simulated in the link and the corresponding time intervals of the system/$Node_1$ and prediction models are identified. The disconnection number is seen in the first column whereas the disconnection time interval is seen in the second column of the table. The values of the second column can be matched with the corresponding disruption intervals specified in Figure 7.7 (right). The third and fourth column represents the time-lapse of the system's predicted start and end time of disconnection. The fifth and sixth column show the time-lapse of the model's predicted start and end time of disconnection. The results are compared in two scenarios. In the first scenario, the predicted start time of both $Node_1$/system and model are compared whereas, in the second scenario, the predicted end time of both $Node_1$/system and model are discussed.

Parameter	Value
Total Sent Packets	3629
Total Received Packets	2765
Total Missed Packets	864
Total Predicted Missed Packets	622

Table 7.6 Parameters for analysis

The observations for the first scenario for all the five disconnections are as follows. The first disconnection is simulated from T_{11} to T_{21} spanning 100 seconds. $Node_1$ interprets the disconnection at T_{31} by pausing the packet transmission. This occurs after a time-lapse of 338 seconds. It means that the $Node_1$ takes about approximately five minutes to realize the network disconnection. On the contrary, the prediction model identifies the disconnection at time T_{i1} with a time-lapse of 79 seconds from the actual disconnection start time. The result indicates that the prediction model is quicker in identifying the disconnection than the sensitivity of the $Node_1$. Moreover, in the second, third, fourth and fifth disconnection, $Node_1$ does not realize the link disconnection and continues to transmit packets leading to packet loss. On the contrary, the prediction model predicts the disconnection start time after a time-lapse of 100, 58, 66, and 77 seconds respectively. These values highly depend on the average of the difference in time intervals between the received packets. The identification of disconnection at T_{i1} can be used to notify the packet handler at $Node_1$ to stop sending packets to the buffer thereby minimizing the packet loss. Therefore, the prediction model is better than the system in both cases of i) system's identification of the disconnection after a longer time-lapse than the model and ii) system's failure to identify the disconnection.

The observations for the second scenario for all the disconnections are discussed as follows. $Node_1$ interprets the connection establishment after the first disruption at T_{41} by resuming the packet transmission. This occurs after a time-lapse of 476 seconds. On the contrary, the prediction model identifies the connection at time T_{n1} with a time-lapse of 314 seconds from the actual disconnection end time. Therefore, the prediction model recognizes the link re-establishment quicker than the $Node_1$ by 162 seconds (476 - 314). It means that the prediction from the model can be used to resume the transmission of packets earlier. $Node_1$ never predicts the disconnection end time for the second, third, fourth, and fifth disruption, as it fails to identify the start of the disconnection. On the contrary, the prediction model predicts the disconnection end time after a time-lapse of 35, 17, 230, and 61 seconds respectively. This time-lapse interval is large for the first and the fourth disconnection. This could be for two reasons i) the simulation of the disconnection breaks the routes between the two nodes, adding recovery time for the routes to be established again to continue packet transmission and ii) the connection is re-established once the simulation of disconnection ends, however, $Node_2$ does not send any packet for long intervals (after connection re-establishment) thereby increasing the predicted disconnection time. Therefore, it can be re-iterated that the prediction model is better than the system in both cases of i) system identifying the connection establishment after a longer time-lapse than the model and ii) system failing to identify and distinguish disconnection and connection re-establishment.

Metric	Received	Missed	Predicted	Not predicted	Value
True Positive	-	✓	✓	-	558
False Positive	✓	-	✓	-	64
True Negative	✓	-	-	✓	306
False Negative	-	✓	-	✓	89
Precision	89.71%				

Table 7.7 Evaluation metrics for packet loss prediction

The list of lost packets written to a text file at the predicted disconnection end time are analyzed (highlighted by yellow vertical lines in layer 4 of Figure 7.8, Figure 7.9 and Figure 7.10). This text file consists of the necessary packet details required to re-send the packets. The next objective was to find the precision of the model to predict the lost packets. For this, the predicted lost packets are compared with the missed packets during transmission. During the entire experiment, a total of 3629 packets were sent from $Node_1$ to $Node_2$, out of which only 2765 were successfully delivered, indicating a total of number 864 missed packets. The model predicted a total of 622 missed packets. All these parameters and their corresponding values are seen in Table 7.6. Based on these values, the evaluation metrics such as *True Positive*, *False Positive*, *True Negative*, *False Negative*, and *Precision* are calculated. The meaning of these metrics with the given scenario of the experiment are explained in Table 7.7. For example, *True Positive* means that the packets were missed and also predicted by the disconnection model. *False Positive* indicates that the packets were received/delivered but, predicted by the disconnection model. *True Negative* means that packets were received and not predicted by the model whereas *False Negative* indicates that packets were missed and not predicted by the model. As a result, the precision of 89% was attained by the approach in predicting the lost packets.

The time-based anomaly detection model successfully identifies all the five disconnections simulated in the network. The predictions of the model can be used to improve the network awareness of the system. The predictions can improve the robustness of the system based on i) early detection of the link disconnection which can be used to notify the packet handler to pause the transmission of packets to the radio buffer, ii) early identification of link re-establishment to resume the packet transfer, and iii) prediction of the lost packets that can be re-transmitted once the connection is established. However, some limitations of this approach are identified, which opens the discussion for future improvements. The link identification model might not be effective when: i) a false disconnection can be predicted by the model if the packet interception time exceeds the average expected time (even if there exists a link connection between $Node_1$ and $Node_2$). In this case, the list of predicted lost packets will consist of successfully delivered packets. ii) Disconnection with smaller time intervals of 10 - 20 seconds might not be detected and it is highly related to frequency and time interval of the packet interception from the other nodes.

8

Conclusion

Tactical Network (TN) is characterized as *ever-changing* and unpredictable because of low bandwidth, high delay, and frequent disruptions. In this sense, tactical systems which connect the user to the network should facilitate data transmission despite the unpredictable network conditions. To accomplish this, the system should be robust to network changes by reducing the impact on data transmission with a suitable adaptation mechanism to the underlying network conditions. Therefore, this study focuses on the problem of *improving the robustness of the tactical system to network variation and disconnection*. The architecture of TACTICS was used as the baseline for implementation and subsequent testing process. This architecture consists of a hierarchical queuing model of the message, packet, and radio buffer. The state-of-the-art solution was Threshold Shaping to regulate the data flow into the radio from packet queue. This mechanism is not sensitive to network changes and therefore, it is relevant to improve the robustness of this architecture to network variations and disconnection.

To improve the system's robustness to network variation, a dynamic shaping of the data flow in the system was proposed based on parameters from multiple layers using the cross-layer information exchange. In this sense, Internet-Packet-Interval (IPI) was computed using buffer metrics and data rate from SNMP protocol to introduce a delay between the packet transmission. In case of no access to the link data rate, an alternate approach was implemented to compute this metric (link data rate) based on cross-layer exchange. On the other hand, to improve the system's robustness to the network disconnection, the machine learning approach and time-based anomaly detection model were implemented. All the above briefly described models were tested with the following observations. The link data rate computation at the sender was validated against the data rate at the receiver with a maximum deviation of about 1 kbps. It suggested that this method is a good alternative for estimating the data rate to compute IPI if the data rate cannot be retrieved from SNMP protocol. Next, the system's data flow adaptation mechanism was tested for constant and ever-changing data rate conditions. For constant network conditions, IPI was compared with Threshold Shaping and it was found that the former solution

is better in terms of precisely maintaining the data flow around the threshold. The subsequent step was to test IPI with ever-changing network conditions using two network sequences with varying data rates. The results indicated that with IPI, the system was able to dynamically adapt its data flow as a combination of both radio buffer and link data rate. An important point to observe is the sensitivity of IPI to give priority to radio buffer occupancy and data rate. For example, in the initial part of the experiment, priority is given to fill up the buffer and then priority shifts to observe both the buffer and data rate. This resulted in efficient usage of the buffer and sensitive network adaptation.

Moving ahead to test the system's robustness to network disconnection, the machine learning approach, and the time-based anomaly detection model are evaluated. The results of our learning approach did not provide sufficient arguments to explore this approach deeply. Potential improvements to this model for the future are discussed in the next section. On the other hand, the results of time anomaly detection suggested that network awareness can be improved in the system with the following observations i) early detection of the link disconnection can be used to notify the packet handler to pause the transmission of packets to the radio buffer, ii) early identification of link re-establishment to resume the packet transfer, and iii) prediction of the lost packets attained a precision of 89% that can be used for packet re-transmission.

8.1 Future Work

The methodology to improve the robustness of the system to network variation includes the implementation and testing of link data rate computation and *Inter-Packet-Interval* in TACTICS architecture. On the other hand, the link disconnection identification models are implemented outside this architecture but, depends on the features obtained from cross-layer information exchange. Although all the above-mentioned methodologies are tested with TACTICS architecture, they can be applied to other frameworks that have an interface to the radio and cross-layer information exchange. This motivates the potential improvement of the implemented solutions and quantitative comparison by other researchers. The following information highlights some of the potential improvement that was identified from this study. The system's data flow can be shaped based on the differential equation as a function of changes in the network and buffer over time. Potential areas to improve the current version of the machine learning approach are highlighted as follows. The training process can be improved with i) more data samples, ii) including more network features that favor in better decision making of the classification, and iii) the features can be subjected to additional data pre-processing and transformations to increase its importance and reduce the noise. Moreover, the hyperparameters of the models can be varied and different classifiers can be deployed. On the other hand, the time-based anomaly detection can be improved to detect even smaller intervals of link disconnection (10 - 20 seconds), thereby increasing the precision of identifying the lost packets. Moreover, to improve the overall adaptation mechanism of the system, and quantify the robustness of transport protocols over ever-changing conditions, we intend to use multi-layer control loops adapting its metrics to the changing conditions and also reuse previous experiences in tactical networks within

machine learning models. Finally, the simulation of the network model can be extended to consider the other network metrics such as the Received Signal Strength Indicator (RSSI), Signal to Noise Ratio (SNR), latency, Round-Trip Time (RTT) which increases the consideration of other parameters that are subjective to changes in a volatile network.

8.2 Publications

The publications related to the present thesis are as follows:

- Under review: Hanavadi Balaraju, P., Rettore, P. H., Rigolin F. Lopes, R., Maligera Eswarappa, S., and Loevenich, J. Dynamic adaptation of data flow to evaluate the robustness of a tactical system: An exploratory study. In *2020 11th International Conference on Networks of the Future (NoF) (NoF 2020)* (University of Bordeaux, France, Oct. 2020)

- Under review: Lopes, R. R. F., Balaraju, P. H., Rettore, P. H., Eswarappa, S. M., Lovenich, J., and Sevenich, P. Quantifying the robustness of tactical systems over ever-changing data rates. In *ITC 2020 International Teletraffic Congress (ITC)* (2020)

- Under review (second round): Lopes, R. R. F., Balaraju, P. H., Rettore, P. H., and Sevenich, P. Queuing over ever-changing communication scenarios in tactical networks. *IEEE Transactions on Mobile Computing* (2020)

- Lopes, R. R. F., Balaraju, P. H., Silva, A. T., Rettore, P. H., and Sevenich, P. Experiments with a queuing mechanism overever-changing datarates in a VHF network. In *IEEE Military Communications Conference (MILCOM)* (Norfolk VA, USA, November 2019)

- Lopes, R. R. F., Balaraju, P. H., and Sevenich, P. Creating ever-changing QoS-constrained dataflows in tactical networks: An exploratory study. In *International Conference on Military Communications and Information Systems (ICMCIS)* (Budva, Montenegro, May 2019)

- Lopes, R. R. F., Balaraju, P. H., and Sevenich, P. Creating and handling ever-changing communication scenarios in tactical networks. In *15th International Conference on the Design of Reliable Communication Networks (DRCN)* (Coimbra, Portugal, March 2019)

Bibliography

[1] TCP Dump and libpcap library. `http://www.tcpdump.org/`, 2019. [Online; September 30, 2019].

[2] Ahrenholz, J. Comparison of core network emulation platforms. In *IEEE Military Communications Conference (MILCOM)* (Oct 2010), pp. 166–171.

[3] Ahrenholz, J., Danilov, C., Henderson, T. R., and Kim, J. H. Core: A real-time network emulator. In *IEEE Military Communications Conference (MILCOM)* (Nov 2008), pp. 1–7.

[4] Amin, R., Ripplinger, D., Mehta, D., and Cheng, B.-N. Design considerations in applying disruption tolerant networking to tactical edge networks. *IEEE Communications Magazine 53*, 10 (2015), 32–38.

[5] Barz, C., Fuchs, C., Kirchhoff, J., Niewiejska, J., and Rogge, H. OLSRv2 for community networks: Using directional airtime metric with external radios. *Computer Networks 93, Part 2* (2015), 324 – 341.

[6] Barz, C., Fuchs, C., Kirchhoff, J., Niewiejska, J., and Rogge, H. Extending olsrv2 for tactical applications. In *International Conference on Military Communications and Information Systems (ICMCIS)* (May 2016), pp. 1–8.

[7] Benincasa, G., Bunch, L., Casini, E., Lenzi, R., Morelli, A., Paulini, M. S., Suri, N., and Uszok, A. Bridging the gap between enterprise and tactical networks via mission-and network-sensitive adaptation. In *IEEE International Conference on Military Communications and Information Systems (ICMCIS)* (2018), IEEE, pp. 1–8.

[8] Cerar, G., Yetgin, H., Mohorčič, M., and Fortuna, C. Machine learning for link quality estimation: A survey. *arXiv preprint arXiv:1812.08856* (2018).

[9] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and Weiss, H. Delay-tolerant networking architecture. RFC 4838, IETF, April 2007. `http://www.rfc-editor.org/rfc/rfc4838.txt`. `http://www.rfc-editor.org/rfc/rfc4838.txt`.

[10] Chen, T., Eswaran, S., Kaplan, M. A., Samtani, S., Shur, D., Sucec, J., and Wong, L. Enhancing application performance with network awareness in tactical networks. In *IEEE Military Communications Conference (MILCOM)* (2011), IEEE, pp. 1158–1163.

[11] Council, N. A. NATO Unclassified C3 Taxonomy Baseline. https://www.nato.int/nato_static_fl2014/assets/pdf/pdf_2019_09/20190912_190912-C3-Taxonomy-baseline.pdf, 2019. [Online; accessed 16-July-2019].

[12] Diefenbach, A., Lopes, R. R. F., Lampe, T. A., Prasse, C., Śliwa, J., Goniacz, R., and Viidanoja, A. Realizing overlay Xcast in a tactical service infrastructure: An approach based on a service-oriented architecture. In *IEEE International Conference on Military Communications and Information Systems (ICMCIS)* (May 2018), pp. 1–8.

[13] Doshi, S., Lee, U., Bressler, B., Bagrodia, R., DiGennaro, M., Oleksa, J., and Chen, Y. Operationally realistic testing of network centric tactical applications in a lab environment. In *IEEE Military Communications Conference (MILCOM)* (2012), IEEE, pp. 1–6.

[14] Dr. Randal Olson ,Dr. Jason H. Moore. Tpot automated machine learning, 2016. https://epistasislab.github.io/tpot/.

[15] Dsouza, M. B., and Manjaiah, D. Congestion free and bandwidth aware multipath protocol for manet. In *2019 1st International Conference on Advances in Information Technology (ICAIT)* (2019), IEEE, pp. 267–270.

[16] Fronteddu, R., Morelli, A., Casini, E., Suri, N., Jalaian, B., and Sadler, L. A content and context-aware solution for network state exchange in tactical networks. In *IEEE Military Communications Conference (MILCOM)* (2017), IEEE, pp. 430–435.

[17] Fronteddu, R., Morelli, A., Mantovani, M., Ordway, B., Campioni, L., Suri, N., and Marcus, K. M. State estimation for tactical networks: Challenges and approaches. In *IEEE Military Communications Conference (MILCOM)* (2018), IEEE, pp. 1042–1048.

[18] Fronteddu, R., Morelli, A., Tortonesi, M., Suri, N., Stefanelli, C., Lenzi, R., and Casini, E. Ddam: Dynamic network condition detection and communication adaptation in tactical edge networks. In *IEEE Military Communications Conference (MILCOM)* (2016), IEEE, pp. 970–975.

[19] Ghosh, A., Li, S.-w., Chiang, C. J., Chadha, R., Moeltner, K., Ali, S., Kumar, Y., and Bauer, R. Qos-aware adaptive middleware (qam) for tactical manet applications. In *IEEE Military Communications Conference (MILCOM)* (2010), IEEE, pp. 178–183.

[20] Gnanasekaran, P., and Vibeeth, B. Link breakage time based qos improvement in mobile ad hoc network. In *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]* (2015), IEEE, pp. 1–4.

[21] Hanavadi Balaraju, P., Rettore, P. H., Rigolin F. Lopes, R., Maligera Eswarappa, S., and Loevenich, J. Dynamic adaptation of data flow to evaluate the robustness of a tactical system: An exploratory study. In *2020 11th International Conference on Networks of the Future (NoF) (NoF 2020)* (University of Bordeaux, France, Oct. 2020).

[22] Holtzer, A., In't Velt, R., Drijver, F., Rogge, H., Kirchhoff, J., Barz, C., Van Adrichem, N., and Hauge, M. Tactical router interoperability: Concepts and experiments. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)* (2018), IEEE, pp. 647–654.

[23] Johnsen, F. T., Bloebaum, T. H., Calero, J. M. A., Wang, Q., Nightingale, J., Manso, M., and Jansen, N. Ws-notification case study and experiment. In *2017 International Conference on Military Communications and Information Systems (ICMCIS)* (2017), IEEE, pp. 1–8.

[24] Khabbaz, M. J., Assi, C. M., and Fawaz, W. F. Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges. *IEEE Communications Surveys & Tutorials 14*, 2 (2011), 607–640.

[25] Lahyani, I., Makki, W., and Chassot, C. Failure prediction for publish/subscribe system on manet. In *2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises* (2012), IEEE, pp. 98–100.

[26] Lampe, T., Prasse, C., Diefenbach, A., Ginzler, T., Sliwa, J., and McLaughlin, S. Tactics tsi architecture. In *International Conference on Military Communications and Information Systems ICMCIS* (2016).

[27] Lindquister, J. J., Johnsen, F. T., and Bloebaum, T. H. Proxy pair optimizations for increased service reliability in dil networks. In *IEEE Military Communications Conference (MILCOM)* (2017), IEEE, pp. 870–875.

[28] Lopes, R. R. F., Balaraju, P. H., Rettore, P. H., Eswarappa, S. M., Lovenich, J., and Sevenich, P. Quantifying the robustness of tactical systems over ever-changing data rates. In *ITC 2020 International Teletraffic Congress (ITC)* (2020).

[29] Lopes, R. R. F., Balaraju, P. H., Rettore, P. H., and Sevenich, P. Queuing over ever-changing communication scenarios in tactical networks. *IEEE Transactions on Mobile Computing* (2020).

[30] Lopes, R. R. F., Balaraju, P. H., and Sevenich, P. Creating and handling ever-changing communication scenarios in tactical networks. In *15th International Conference on the Design of Reliable Communication Networks (DRCN)* (Coimbra, Portugal, March 2019).

[31] Lopes, R. R. F., Balaraju, P. H., and Sevenich, P. Creating ever-changing QoS-constrained dataflows in tactical networks: An exploratory study. In *International Conference on Military Communications and Information Systems (ICMCIS)* (Budva, Montenegro, May 2019).

[32] Lopes, R. R. F., Balaraju, P. H., Silva, A. T., Rettore, P. H., and Sevenich, P. Experiments with a queuing mechanism overever-changing datarates in a VHF network. In *IEEE Military Communications Conference (MILCOM)* (Norfolk VA, USA, November 2019).

[33] Lopes, R. R. F., Viidanoja, A., Lhotellier, M., Diefenbach, A., Jansen, N., and Ginzler, T. A queuing mechanism for delivering qos-constrained web services in tactical networks. In *2018 International Conference on Military Communications and Information Systems (ICMCIS)* (2018), IEEE, pp. 1–8.

[34] Lund, K., Eggen, A., Hadzic, D., Hafsoe, T., and Johnsen, F. T. Using web services to realize service oriented architecture in military communication networks. *IEEE communications magazine 45*, 10 (2007), 47–53.

[35] Małowidzki, M., Kaniewski, P., Matyszkiel, R., and Bereziński, P. Standard tactical services in a military disruption-tolerant network: Field tests. In *IEEE Military Communications Conference (MILCOM)* (2017), IEEE, pp. 63–68.

[36] Marcus, K., Barz, C., Kirchhoff, J., Rogge, H., Nilsson, J., in 't Velt, R., Suri, N., Hansson, A., Sterner, U., Hauge, M., Lee, K., Holtzer, A., Buchin, B., Peuhkuri, M., and Misirlioglu, L. Evaluation of the scalability of OLSRv2 in an emulated realistic military scenario. In *IEEE International Conference on Military Communications and Information Systems (ICMCIS)* (May 2017), pp. 1–8.

[37] Marsden, S., and Vankka, J. Tactical network modeller simulation tool. In *IEEE Military Communications Conference (MILCOM)* (Oct 2015), pp. 1087–1092.

[38] Miao, Y., Sun, Z., Wang, N., and Cruickshank, H. Comparison studies of manet-satellite and manet-cellular networks integrations. In *2015 International Conference on Wireless Communications & Signal Processing (WCSP)* (2015), IEEE, pp. 1–5.

[39] Moore, S., Amin, R., Ripplinger, D., Mehta, D., and Cheng, B.-N. Performance evaluation of a disruption tolerant network proxy for tactical edge networks. In *IEEE Military Communications Conference (MILCOM)* (2016), IEEE, pp. 964–969.

[40] Morelli, A., Stefanelli, C., Tortonesi, M., Lenzi, R., and Suri, N. A proxy gateway solution to provide qos in tactical networks and disaster recovery scenarios. In *Proceedings of the 11th ACM Symposium on QoS and Security for Wireless and Mobile Networks* (2015), ACM, pp. 43–50.

[41] Nanda, S., Tao, C., Firoiu, V., Zeng, H., Ahn, G.-S., and Deng, J. Cross-layer tcp adaptation in disco for tactical edge networks. In *MILCOM 2016-2016 IEEE Military Communications Conference* (2016), IEEE, pp. 958–963.

[42] Peng, A. S., Moen, D. M., He, T., and Lilja, D. J. Automatic dynamic resource management architecture in tactical network environments. In *IEEE Military Communications Conference (MILCOM)* (2009), IEEE, pp. 1–7.

[43] Phemius, K., Seddar, J., Bouet, M., Khalifé, H., and Conan, V. Bringing SDN to the edge of tactical networks. In *IEEE Military Communications Conference (MILCOM)* (Nov 2016), pp. 1047–1052.

[44] Poylisher, A., Sultan, F., Ghosh, A., Li, S.-w., Chiang, C. J., Chadha, R., Moeltner, K., and Jakubowski, K. Qam: A comprehensive qos-aware middleware suite for tactical communications. In *IEEE Military Communications Conference (MILCOM)* (2011), IEEE, pp. 1586–1591.

[45] Ruffieux, S., Gisler, C., Wagen, J., Buntschu, F., and Bovet, G. Take — tactical ad-hoc network emulation. In *IEEE International Conference on Military Communications and Information Systems (ICMCIS)* (May 2018), pp. 1–8.

[46] Sakthivel, M., Gnanaprakasam, T., and Rao, K. S. K. Reliable data delivery in manets using pgf and vh scheme. In *2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE)* (2017), IEEE, pp. 1–6.

[47] Scott, K., Refaei, T., Trivedi, N., Trinh, J., and Macker, J. P. Robust communications for disconnected, intermittent, low-bandwidth (DIL) environments. In *IEEE Military Communications Conference (MILCOM)* (Nov 2011), pp. 1009–1014.

[48] Selvakumar, G., Ramesh, K., Ungati, S., and Chaudhari, S. Ip packet forwarding mechanism in multi-hop tactical wireless networks. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (2018), IEEE, pp. 547–551.

[49] Shen, J.-j., Gan, Z.-c., and Yu, X.-h. The research on tactical internet simulative training and its dynamic real-time network simulation. In *2nd IEEE International Conference on Information Management and Engineering* (2010), IEEE, pp. 354–358.

[50] Srinivasan, S. M., Truong-Huu, T., and Gurusamy, M. Machine learning-based link fault identification and localization in complex networks. *IEEE Internet of Things Journal 6*, 4 (2019), 6556–6566.

[51] Suri, N., Fronteddu, R., Cramer, E., Breedy, M., Marcus, K., i. '. Velt, R., Nilsson, J., Mantovani, M., Campioni, L., Poltronieri, F., Benincasa, G., Ordway, B., Peuhkuri, M., and Rautenberg, M. Experimental evaluation of group communications protocols for tactical data dissemination. In *IEEE Military Communications Conference (MILCOM)* (Oct 2018), pp. 133–139.

[52] Suri, N., Fronteddu, R., Cramer, E., Breedy, M., Marcus, K., in't Velt, R., Nilsson, J., Mantovani, M., Campioni, L., Poltronieri, F., et al. Experimental evaluation of group communications protocols for tactical data dissemination. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)* (2018), IEEE, pp. 133–139.

[53] Suri, N., Hansson, A., Nilsson, J., Lubkowski, P., Marcus, K., Hauge, M., Lee, K., Buchin, B., Misirhoğlu, L., and Peuhkuri, M. A realistic military scenario and emulation environment for experimenting with tactical communications and heterogeneous networks. In *IEEE International*

Conference on Military Communications and Information Systems (ICMCIS) (2016), IEEE, pp. 1–8.

[54] Suri, N., Nilsson, J., Hansson, A., Sterner, U., Marcus, K., Misirlioğlu, L., Hauge, M., Peuhkuri, M., Buchin, B., In'T Velt, R., et al. The angloval tactical military scenario and experimentation environment. In *IEEE International Conference on Military Communications and Information Systems (ICMCIS)* (2018), IEEE, pp. 1–8.

[55] Whitbeck, J., Lopez, Y., Leguay, J., Conan, V., Rosenberg, O., and Tessier, O. Using UHF connectivity to off-load VHF messaging in tactical manets. In *IEEE Military Communications Conference (MILCOM)* (Nov 2011), pp. 961–966.

List of Figures

List of Tables

A

Appendix

A.1 Link disconnection simulation

Disconnection of the link is caused using the hardware devices shown in Figure A.1. This model was developed by the authors in [32] as a part of their future work. The figure shows the Raspberry-Pi (A) and the Relay (B). The Relay is connected to the coaxial cable of the link between the radios. An Adjustable Converter Module is used for step-up boost power supply that converts 5v to 12v; necessary to support the relay. In addition, there is a simple circuit with a transistor and resistor to open/close the electric pulse and activate the converter module (inside the Raspberry-Pi box). Moreover, an attenuator is used in order to interrupt the sign by the relay. The Raspberry-Pi acts as controller to deactivate/activate the relay in a specified time interval, causing disconnection/connection respectively.

With the use of the hardware devices highlighted in Figure A.1, a python socket program is implemented on the server side to listen to disconnection requests. This is explained in Algorithm 4. In step 1, the channel numbers are used for numbering the General Purpose Input/Output (GPIO) pins. In step 2, the 23^{rd} pin is set for output. In step 3 and 4, a socket is bound to the Raspberry port and listens for connection requests from the client. The implementation of the client side to request for disconnection is explained in Algorithm 5. If there has been a request (step 5), a connection is established in step 6. If the connection establishment was successful (step 7), then *data* is received from the client in step 8. The *data* corresponds to the time interval specification for the link disconnection. The time interval is converted to integer format ($timeInt$) in step 9 and the current time is noted as the start time of the disconnection ($startTime$) in step 10. GPIO is set high to start the relay, which causes the cable disconnection (step 11). The 16^{th} pin port in the assembly architecture is GPIO23. The disconnection is continued in step 12 for the specified time interval($timeInt$). GPIO is set low to stop the relay, which causes the connection re-establishment in step 13. The current time is marked the end of the disconnection time ($endTime$) in step 14. The time interval of the disconnection is

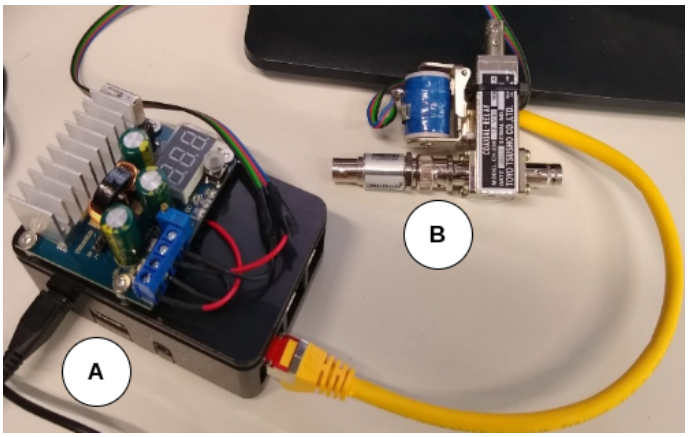

Figure A.1 Raspberry pi to implement disconnection

```
1: GPIO.setmode(GPIO.BCM)
2: GPIO.setup(23, GPIO.OUT)
3: socket.bind (raspberry.host,raspberry.port)
4: socket.listen ()
5: while True do
6:     connection = socket.connect()
7:     while True do
8:         data = connection.recv(1024)
9:         timeInt = int.fromBytes(data, "big")
10:        startTime = time.time()
11:        GPIO.output(23, GPIO.HIGH)
12:        time.sleep(timeInt)
13:        GPIO.output(23, GPIO.LOW)
14:        endTime = time.time()
15:        disruptionTime = endTime - startTime
16:        connection.sendall(data))
17:    end while
18: end while
```

Algorithm 4 Server side to implement disconnection

```
1: s = socket.socket(socket.AFINET, socket.SOCKSTREAM)
2: s.connect((ipAddress, portNumber))
3: byteFormat = disconnectionTime.toBytes(2, big)
4: s.sendall(byteFormat)
5: data = s.recv(1024)
```

Algorithm 5 Client side to implement disconnection

calculated in step 15 ($endTime - startTime$). This information is sent back to the client in step 16.

At the client side, a python socket program is implemented to request for disconnection of the cables and it is explained with reference to Algorithm 5. An INET socket is created in step 1. This socket is used to connect to the server using the IP address and the port number of the server in step 2. The time interval of the disconnection is

converted from integer to byte format ($byteFormat$) in step 3. The converted time interval in byte format is sent to the server for implementing disconnection in step 4. After the simulation of disconnection, the client receives data from the server in step 5.

Acronyms

ACM Agile Computing Middleware. 15

AE Adaptation Engine. 14

Auto-DRM Automatic Dynamic Resource Management. 19

BAAINBw Bundesamt für Ausrüstung, Informationstechnik und Nutzung der Bundeswehr. 16

BFT Blue Force Tracking. 8, 20, 34

BLOS Beyond Line of Sight. 10

C2 Command and Control. 6

C3 Consultation, Command, and Control. 5–8, 16, 67

CI Control Interface. 14

CIS Communication and Information Systems. 7

COI Community of Interest. 7–9, 16

CORE Common Open Research Emulator. 22

DDAM Dynamic Detect and Adapt Mechanism. 19

DISCO Distributed Cross-layer Architecture for Network Awareness and Opportunistic Transport. 19

DTG Dynamic Throughput Graph. 14, 19

DTN Disruption Tolerant Networks. 10, 18, 19

EMANE Extendable Mobile Ad-hoc Network Simulator. 21, 22

ETC Enhanced Tactical Computers. 8

FFT Friendly Force Tracking. 34

GPIO General Purpose Input/Output. 71

HF High Frequency. 15

IMSBridge Information Management Services Bridge. 19

IP Internet Protocol. 16, 18, 27, 32–35, 72

IPI Internet-Packet-Interval. v, 29, 35, 45, 47, 48, 57, 58

JENM JTRS Enterprise Network Manager. 21

JNE Joint Network Emulator. 21

LAN Local Area Network. 10, 15

LOS Line of Sight. 10

MAN Metropolitan Area Network. 10

MaNaTIM Mission and Network Adaptive Tactical Information Management. 19

MANET Mobile Adhoc Network. v, 1, 10

MTU Maximum Transfer Unit. 19

NAS Network Awareness Service. 14, 19

NATO North Atlantic Treaty Organization. 6, 16

NB Narrow Band. 22

NMS Network Management System. 15

NTP Network Time Protocol. 38

OLSR Optimized Link State Routing. 16, 19, 22, 24, 26, 27, 36

OneSAF One Semi-automated Force. 21

OPNET Optimized Network Engineering Tool. 21

OSI Open Systems Interconnection. 21

PDR Packet Delivery Ratio. 14

QAM QoS-aware Adaptive Middleware. 19

QoS Quality of Service. 9, 14, 19, 20

RF Radio Frequency. 2, 12, 21, 24

RSSI Received Signal Strength Indicator. 26, 59

RTT Round-Trip Time. 59

S2ES Smart Status Exchange Service. 16, 19

SatCom Satellite Communications. 10, 15

SENSEI Smart Estimation of Network Information. 19

SNMP Simple Network Management Protocol. 15, 19, 26, 27, 29, 36, 38, 39, 45, 57

SNR Signal to Noise Ratio. 26, 59

SOA Service-Oriented Architecture. 9

SVN Software Virtual Networks. 21

TACTICS Tactical Service-Oriented Architecture. 3, 13, 16, 19, 25, 26, 31, 35, 36, 39, 45, 57, 58

TCP Transmission Control Protocol. 14, 15, 18, 19, 31

TN Tactical Network. v, 1, 57

UDP User Datagram Protocol. 16, 18, 31

UDT UDP-based Data Transfer. 14, 19

UHF Ultra High Frequency. 10

VHF Very High Frequency. 10

WAN Wide Area Network. 10

WB Wide Band. 22

WS Web Services. 9

WTD-81 Wehrtechnische Dienststelle für Informationstechnologie und Elektronik. 16