Master's Thesis

Design of QoS-constrained Dataflows to Test Store-and-forward Mechanisms in Tactical Networks

Author:
Adrián Kevin Toribio Silva
University of Bonn, Institut für Informatik
Rheinische Friedrich-Wilhelms-Universität Bonn
Email: adrian.toribio.silva@alumnos.upm.es

Advisor:
Dr. Roberto Rigolin F. Lopes
Co-Advisors:
Dr. Paulo Henrique L. Rettore and Dr. Matthias Frank

Research Scientists, Fraunhofer FKIE, Bad Godesberg, Germany
Email: [roberto.lopes,paulo.lopes.rettore]@fkie.fraunhofer.de

November 8, 2020

Design of QoS-constrained Dataflows to Test Store-and-forward Mechanisms in Tactical Networks

Adrián Kevin Toribio Silva

Abstract

Tactical networks, as the basis for military deployments, must deal with limitations such as low bandwidth, high delay and frequent disruptions during communications. To ensure robust and secure transmissions that do not jeopardize the loss of information, mechanisms such as store-and-forward implemented in tactical systems or middlewares, among others, are used. These solutions make use of algorithms that control the filling of radio buffers and the processing of packet queues using cross-layer information exchange. In order to improve and test this type of store-and-forward mechanisms it is required to make use of scenarios close to reality, where both the flow of data generated by users and the conditions of the network are changing. Our work focuses on designing QoS-constrained dataflows through a tool to provide realistic situations that help test and improve tactical systems. For this purpose, the term randomness is included, which is not present in many of the works of literature that use artificial flows. The three experiments carried out simulate different traffic on a VHF network with which to test the operation of the tool, analyze the limits of TSI middleware performance and the impact that saturation of a network has on the quality of communications. In the first one the effect of sending a large message (1000 KB) is discussed and in the other two tests 1000 messages of 1 KB are sent in burst mode and with time-windows respectively. The results are then compared in terms of the evolution of the radio buffer, IP packets sent and received and Inter-Packet Interval (a concept introduced in [1]).

Keywords QoS-constrained dataflows, tactical networks, store-and-forward mechanism, ever-changing communication scenarios

Acknowledgements

Now that I close this stage I would like to dedicate a few lines to all the people who have accompanied me.

First of all, I would like to thank Roberto for the opportunity to do this master thesis. His dedication, support and ambition have undoubtedly marked the development of this work. I would also like to thank the help received from the team, especially from Pooja and Paulo, for guiding me in the right direction, providing ideas to keep moving forward and assisting me when I needed it. The good atmosphere we have had during these months has made my work bearable and entertaining.

I could not fail to mention Paula and Miguel, with whom I have shared, suffered and lived each and every one of the experiences of this Erasmus. We finally did it! I do not doubt that you will leave your mark wherever you work.

And last but not least, I would like to thank my family for all their affection and attention. That in spite of the distance they were always by my side to know how my progress was going. I love you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Tactical networks are the base of military deployments, allowing the establishment of communication between entities in tactical scenarios, such as people, vehicles, base stations and others. The use of these means of communication is essential to establish secure and reliable communications in emergency situations such as warfare, natural disasters or evacuations. In addition to provide capabilities to exchange information (pictures, text, regular intervals, etc.) among military tactical forces, it also enables the adoption of Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance (C4ISR) applications [4].

These networks are complex environments that must treat confidential information and events by handling the use of a diversity of nodes from static sensors to servers mounted in mobile vehicles (drones, robots, etc.) and different communication technologies (e.g Very High Frequency (VHF), Ultra High Frequency (UHF) and Satellite Communications (SatCom)). Therefore tactical networks differ from commercial networks in their rate constraints, anti-jamming needs, mobility/dynamic topology, low bandwidth, high latency and the adversary's activity (e.g. jamming) [5].

In a context where the user-behaviour and the network conditions are unpredictable and have a changing nature, a tactical network can be divided into three different problems as it is referred in our study and showed in Figure 1.1. Problem A involves the user-behaviour where the military forces can use different Command and Control (C2) services to exchange information. Here there is a wide variety of information types (verbal, written or visual), ways to send it and Quality of Service (QoS) parameters of the messages. Problem B deals with the effects of the network constraints in data transmissions. Due to the coexistence of different technologies with different specifications (bandwidth, reliability, datarate, etc.), possible natural barriers (mountains, weather, forests, etc.) and nodes in movement the QoS may be affected. Finally, problem $A|B$ is located in the tactical system layer whose main task is to connect the user layer with the communication layer, making its intervention as a mediator opaque for both tiers and adapting the system to the changing conditions.

The store-and-forward mechanisms are implemented in the tactical system (also called tactical middleware) to deal with delays and disruptions. These techniques are therefore responsible for preventing the loss of traffic packages in their transmission and their role is especially important in emergency cases, where bottlenecks can be expected. In this sense, they use algorithms that decide how and when to fill

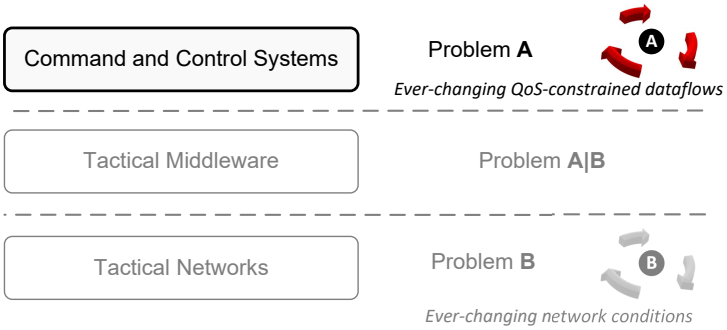


Figure 1.1: The three problems: A, B and $A|B$ [1]

the radio buffers and how to treat queued packets using cross-layer information exchange.

The North Atlantic Treaty Organization (NATO) alliance [6] is responsible to guarantee the freedom and protect the stability of its affiliates through political and military means, and they have an annual program called NATO Coalition Warrior Interoperability eXploration, eXperimentation, eXamination, eXercise) (CWIX). CWIX allows nations to experiment, test and reduce the risk of their systems before undertaking missions [7]. One of the European Defence Agency on-going projects in the field of tactical networks is called Tactical Service-Oriented Architecture (TACTICS) [8, 9] and it aims to improve situation awareness in tactical environments.

Concerning the NATO CWIX program, the results of TACTICS's have been evolved over the last years using communication scenarios that resemble reality and to stress the performance bounds of tactical systems. At this point one of the existing research fields is the analysis and testing of tactical middlewares based in web services combining military radios and the multinational interoperability tests at Future Core Services (FCS) [10].

The present thesis is part of TACTICS and addresses the changing nature of user-behaviour based on evolving the results of the following Lopes et al. works [1, 11, 10]. Imposed by the VHF radios constraints in coverage ($\sim$ 20 km) and nominal datarate (9.6 kbps) Lopes et al. [1] propose a hierarchical queueing mechanism to control and monitor radio buffers while delivering web services in tactical networks. To deploy QoS-constrained (e.g. priority, reliability and time of expiry) web services in this type of network, several researchers [12, 13, 14, 15, 16, 17, 18, 19] suggest the use of middlewares following the NATO's C3 taxonomy [2]. The middleware used in the experiments reported in this thesis is called Tactical Service-Oriented Infrastructure (TSI), it uses User Datagram Protocol (UDP) transport protocol and it controls the data flow by monitoring the radio buffer. The core services are mainly based on two different queues (message queue Q_m and packet queue Q_p), a *QoS handler* and a control loop relying on the cross-layer information exchange between the two queues and the radio buffer.

Considering the aforementioned literature our investigation focuses on Problem A as shown in Figure 1.1, aiming to create a framework (tool) to simulate the user's behavior at a particular mission in order to evaluate the store-and-forward

mechanisms in tactical networks. This can help other researchers to analyze the efficiency of their systems stressing the performance bounds of them by incrementally generating more challenging QoS-constrained dataflows.

1.1 Motivation

The 2005 NATO Network Enabled Capability Feasibility Study (NNEC FS) issued two important recommendations related to the interactions implied in military operations [20]. As a consequence, a Service Oriented Architecture (SOA) based on web service standard was suggested. Lund et al. [21] raised the issue of implementing the information infrastructure as SOA by using the IP network protocol. SOA enables to provide and use applications/resources as services, allowing rapid application integration, multi channel access to applications and dynamic information sharing [22]. In Network Enabled Capabilities (NEC) the major defiance is to establish an exchange-information channel among military units that can operate over disadvantaged grids (with frequent disconnections, low data rate and high delay). Web services are based on Extensible Markup Language (XML) standards and Simple Object Access Protocol (SOAP) and they have an ample amount of information overhead, consequently, they are not very efficient in disadvantage grids. Therefore the authors suggested some techniques and mechanisms that contribute to the use of web services in disadvantaged grids: (1) GZIP and binary XML as compression methods, (2) the use of S4406 instead of Hypertext Transfer Protocol (HTTP) as transport for SOAP messages, (3) adding proxies to the system or (4) optimization of message representation and content.

The use of web services allows offering applications as services in a rapid easy way to integrate for example obstacle or positioning alerts. The middleware acts as an intermediary between web services and tactical networks, handling ever-changing communication scenarios. Adapting web services to tactical networks means dealing with challenges that vary from the mobility of the nodes to degraded conditions of the communications [4]. Due to the lack of network infrastructure support, the adversary's actions, frequent interference and disruption, high latency and low datarates the design choices become complicated. These limitations have revealed three great needs in the development of effective middlewares and applications: (1) the inclusion of randomness in the user behaviour (differing QoS-constrained dataflows), (2) network conditions (varying among different network states) and (3) how to manage the user data-flow given the current network circumstances.

Creating a close to real communication scenario where the user behaviour and network states varying independently over time can contributes to improving the deployment in dynamic and resource constrained environments. Furthermore, in simulation scenarios, the precision/overhead relationship in the network must be balanced and difficulties can be found with security policies when controlling traffic and characterizing the information. However, the experiments carried out in the literature suggest solutions tested with specific user behaviors and limited messages, which lacks the element of surprise (randomness). This indicates that the efficiency of the system is not tested against a real dynamic nature of user behavior (e.g. varying QoS-constrained data-flows) suggesting a significant gap between reality and test-bed scenarios, particularly in worst case scenarios.

Therefore, the purpose of our research is to fill this gap addressing the Problem

A previously defined in [11] as following. Considering the fact that mobile nodes in a tactical network are hosting C2 systems where the users are exchanging messages, it coexists messages with different QoS parameters like priority, reliability and Time of Expire (ToE). Thus a realistic combination of these diverse messages must be part of a close to real emulation. The preceding work of Lopes et al. uses Table 1.1 to list five *user services* sorted by their priorities: *Flash 0*, *Immediate 1*, *Priority 2* and *Routine 3*. The predefined message frequency λ is unknown for most of the services as they are stochastic and unpredictable. However, there may be predictable services such as Friendly Force Tracking (FFT), which sends messages with a specific frequency *f*.

Service	λ	**Priority**	**Reliability**	**ToE (sec)**
s_1 Medical evacuation	?	0 Flash	Yes	300
s_2 Obstacle alert	?	1 Immediate	Yes	150
s_3 Picture	?	2 Priority	Yes	3600
s_4 FFT	*f*	3 Routine	No	120
s_5 Text	?	0,1,2,3	Yes/No	?

Table 1.1: Message priority, frequency and time to expire

Thus, the research question is: How to create an ever-changing message' behavior in order to challenge an entire communication system?. Our hypothesis is that *"Varying the message in four different variables such as number, size, time, and priority we are able to simulate the changing nature of user-behavior, and consequently, stressing any communication network."*. In this sense, we focus on include the elements of chance (randomness) and create a wide range of message combinations including the best/worst case scenarios within extreme communication scenarios. Therefore, as QoS-constraints demanded, we suggest randomness in the priorities of the traffic sent and variety in the same in terms of size as well in content and frequency of sending. For this purpose, stochastic models are implemented in our tool to create QoS constrained dataflows as it is explained in the following chapters of this thesis.

1.2 Goals

Focusing on the Problem *A* defined in [11], this thesis proposes the implementation of an application to create a sequence of QoS-constrained messages and test a store-and-forward mechanisms in tactical networks. In such a way that the performance and limit bounds of the system (particularly the store-and-forward mechanism) or any other can be defined in a pragmatic way using quantitative results.

1.2.1 Specific Goals

As the development of the main objective, this study is divided into the following specific goals:

- Conduct a study of the state of the art in the analysis of communication scenarios in tactical networks: through an exhaustive literature review that allows establishing a work plan and thus develop the experiments to generate possible new contributions to the scientific community.

- Given the services in Table 1.1, implement a methodology to create sequences of QoS-constrained messages. These sequences of messages have to vary in four different variables such as number, size, time, and priority.
- Design the experiment with and without (baseline for quantitative comparisons) using a tactical middleware.
- Analyze the results through a statistical and mathematical treatment, discussing them and proposing future directions.

1.3 Contributions

During this thesis, the following publication has been presented:

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

1.4 Outline

The rest of this thesis is organized as follows. In Chapter 2, we analyze the literature review related to communication scenarios in tactical networks. Chapter 3, the TACTICS project is described and the problems related to it are defined. Chapter 4 shows the benchmark solution and Chapter 5 the results and experiments are discussed. Finally, Chapter 6 concludes the thesis and discuss future works.

Chapter 2

Literature Review

This chapter discusses fundamental concepts supporting the development of tactical systems and related investigations that address solutions to simulate QoS-constrained dataflows to test networks. This chapter is organized as follows. First, main concepts in the field of tactical networks are explained. Second, two of the most used message technologies in data transmission are described. Finally, diverse tools developed to test different types of network that can be found in the literature are exposed and compared.

2.1 Fundamental concepts

2.1.1 Command and Control (C2)

Command and Control (C2) is a set of technical, directional and executive processes and attributes employed by an organization to accomplish missions and solve problems making use of information, human and physical resources [24]. It is composed of two different and related functions, command and control. Applied to tactical environments, command is the authority that a commander exercises for efficiently using available resources and for planning, organizing, coordinating, and controlling military systems and forces for the accomplishment of assigned missions. The main objective of control is to manage the mission problem and minimize the risk of not reaching a proper solution and to regularize the battlefield systems and forces according to the commander's intent.

2.1.2 Consultation, Command and Control (C3) Taxonomy

Similar to the description of C2 competences, the C3 capabilities are those related to the NATO's *Consultation, Command and Control* activities and are focused mainly on information sharing and interoperability. Thus, the *C3 Taxonomy* is a principle that describes the concepts and their relationships involved in all the C3 life cycle activities [2]. Providing an environment with a shared language to synchronize these activities and improve connecting NATO's Strategic Concept and Political Guidance as it is shown in 2.1. This taxonomy is used in most of the middlewares for tactical networks proposed in the literature [12, 13, 15, 25, 26].

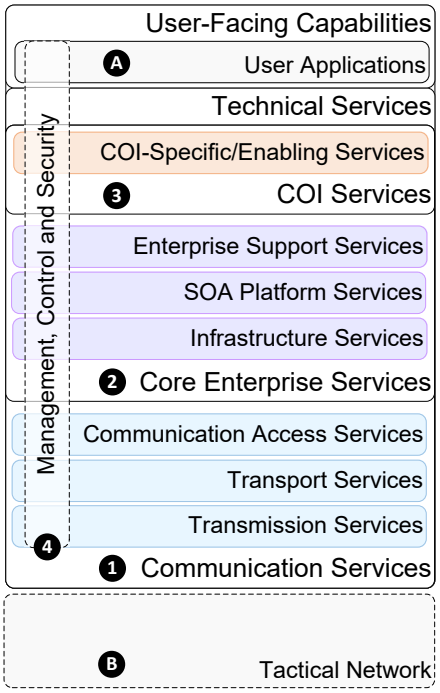


Figure 2.1: Communication and Information System (CIS) Capabilities layer in the C3 Taxonomy [2]

The Communication and Information Systems (CIS) Capabilities layer describes the components of the capabilities required to meet NATO's information system and communication needs. The Back-End Capabilities layer represents the list of services and equipment that is required to enable User-Facing Capabilities, expressing the requirements in communications and data processing [2]. The main blocks that compose these layers are:

A User-Facing Capabilities: also called software applications. They act as a logical interface that enables the user to manage and process data to execute tasks using software components (e.g. Air, Land, Maritime or Space applications). It is therefore related to problem A.

1. Communication Services: interconnection, exchange and access services that are essential to managing a tactical network. As for example Multimedia access, Transport or Communication access services. It is therefore related to problem $A|B$.

2. Core Enterprise Services: provides generic, technical functionality to implement *service-oriented architecture* (SOA). In this block can be found services such as Message-Oriented Middleware, Infrastructure or SOA Platform services. It is therefore related to problem $A|B$.

3. Community of Interest (COI): manages a collective group of users with common missions or business processes. Services such as Modeling and Simulation, Operations Planning or Situational Awareness are included here. It is therefore related to problem $A|B$.

4. Management, Control and Security: cross-layer mechanism to bundle components from various classes of taxonomy into a collection with a particular common characteristic. In this case control, security and service management. It is therefore related to problem $A|B$.

B Tactical Network: the conditions of tactical networks vary over time, changing between connected, disconnected or limited states. It is therefore related to problem B.

2.1.3 Middleware

Middleware is a software located between the kernel of the operating system and the applications. It provides services to software applications to establish communication and administration of the data among them. The use of middlewares allows hiding the processes and complexity involved in connecting the front-end that the user perceives with the back-end that offers the data. In the field of tactical networks, middlewares take an important role because they act as mediators in the adoption of web services in ever changing communication scenarios [27]. Some of the types of middleware are:

- Application server: is a framework that provides the functions to create applications and a server on which they can be executed [28].

- Data integration: the practice of manipulation of heterogeneous data (from different sources) in a unified view, so that users can access and manipulate them [29, 30].

- Application integration: it involves combining data from different applications through an integration framework [31].

- Application Programming Interface (API): sets of tools, definitions and protocols for designing application software, which allows a product or service to communicate with other products and services [32].

- Message-Oriented Middleware (MOM): it supports the exchange of general-purpose messages in a distributed application environment. Ensuring the delivery by using reliable queues and by providing security, and administrative services required when the destination node is slow or busy (e.g. [8] discussed in details in the next chapter, Chapter 3).

2.1.4 Java Message Service vs Web service

An interface-based architecture is needed to access the applications and services that an organization has integrated. This interface-based architecture includes such technologies as JMS or web services [3]. To be able to discuss the message benchmarks with greater rigour the characteristics of two of the most used technologies are described and summarized in Table 2.1. Most of the benchmarks found in the literature are based on JMS whereas our proposal is based on web services.

Java Message Service (JMS)

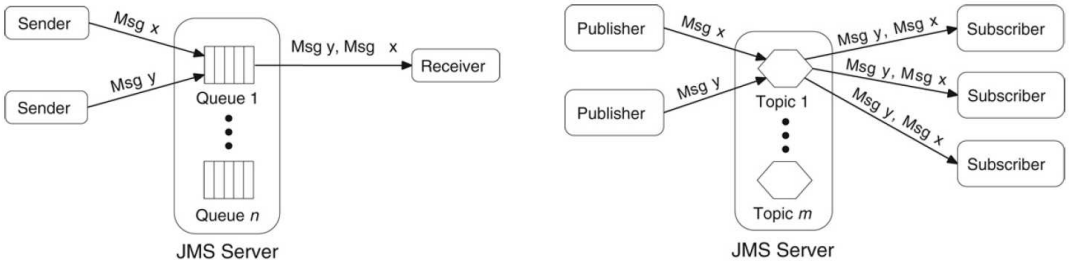
The JMS is the *de facto* industry standard interface for MOM [33]. It is an asynchronous message-based interface that supports two messaging models illustrated in Fig. 2.2, point to point and publish/subscribe. In point to point there are two clients, a sender and a receiver (1:1). This model ensures the arrival of the message because messages are sent and stored in a First in, First out (FIFO) queue. In the case of publish/subscribe there are several clients, some who publish topics and those who subscribe to these topics and will receive any updates related to them. Unlike the point-to-point model, this model tends to have more than one consumer (1:n). This is also called one-to-many message exchange pattern, which is common in tactical networks given the hierarchical structure of military operations.

JMS frameworks also offer the possibility to work simultaneously (sender and receiver) in a simulating synchronous mode. Furthermore, their structure guarantees the interoperability between different frameworks and the delivery of messages. The delivery modes are:

Non-Persistent/Persistent: in the persistent mode the messages are logged to persistent storage such as a database or a file system whereas in non-persistent messages they are stored in memory buffers that could be lost in case of a server crash.

Non-Transactional/Transactional: messages can be send as a transaction or not. A transaction is a set of messaging operations that are executed as an atomic unit of work.

Non-Durable/Durable: subscriptions can be durable or non-durable. In the Non-durable case a subscriber will only receive published messages while it is active. In contrast, subscribers do not loose any message during inactivity cycles if they have a durable subscription.



(a) Point to point messaging (b) Publish/subscribe messaging

Figure 2.2: Point to point and publish/subscribe messaging models

Web Services (WS)

Web services are an implementation of SOA. In SOA there exist three important components: (1) provider, (2) broker and (3) requester. Web services are interfaces between a provider and a requester [21]. In that context, the service provider publishes the services that are not directly visible to the requesters and are managed

by the broker. The consumers may know the protocol to subscribe to the interested service and how to invoke it by using the broker. Service consumers do not know the way that the services are implemented.

Web services can perform either in a synchronous request/reply mode or in an asynchronous mode and are based on XML. Some of the standards used are: (1) SOAP for information transmission, (2) Web Services Description Language (WSDL) as the protocol for bindings and message formats and (3) Universal Description Discovery and Integration (UDDI) that is a registry for available web services.

	Java Message Service (JMS)	**Web Services (WS)**
Interface coupling	No (Payload agnostic)	Yes
Technical coupling	Yes	No
Portability	No (Java technology only)	Yes (Multi-language)
Reliability	Yes	HTTP-R binding for SOAP
Transactional Support	Limited in scope Only to the queue entry point	Future
Security	Not part of the standard	Limited to SOAP SOAP-SEC
Synchronous mode	Do it yourself	Yes Major use
Asynchronous mode	Yes	Yes (Document oriented interface)
Event-driven, push mode	Yes	Yes

Table 2.1: Comparison between JMS and Web services [3]

2.2 Message Benchmarks

Message benchmarks tests are used in a component or an entire end to end Information Technology (IT) system to determine the performance characteristics of the system or application. The main goal is to stress the system to determine its limitations and bounds to guarantee that applications meet their QoS requirements. Therefore, we focus our study on the different types of benchmark which are explained below.

Several works are related to the performance of JMS technology in MOM middlewares, showing different points of view and levels of detail. To achieve this they make use of benchmarks that test different JMS servers in order to compare their capacities. Some of the vendors provide their frameworks, for instance, JBoss Messaging Performance Framework [34], Apache's ActiveMQ JMeter Performance Test [35] and MQ-IBM CPH Performance Harness [36]. The main drawback of this group is that they were created to stress specific MOM features and are not always valid for analyse the overall performance. Furthermore, the generated workload is synthetic and consequently they miss the element of randomness, not reflecting a close to real application scenario in tactical networks.

JBoss Messaging is an updated JMS provider version of JBossMQ in the JBoss Enterprise Middleware Stack (JEMS) [37]. It is open source and a standard-based messaging platform that also offers flexibility to SOA initiatives. The simple benchmark used in JBoss is called JBoss Messaging Performance Framework and sends bursts of 1000 KB non-persistent messages gradually increasing the sending rate up to 3000 messages/second. It only supports a point to point model so it is necessary to indicate the receivers. The generated environment is limited in the size of messages and delivery time, in turn, does not offer the possibility to modify the payload of the message or specify a certain priority to create different types. As a result, it

cannot be applied in tactical networks, where different services send messages and the network suffers disconnections which alternatively causes message bursts.

Apache JMeter application is a competitive open source software originally designed for measuring the performance of Web Applications [38]. It can be used via Graphical User Interface (GUI) or command line and it is compatible with a wide variety of applications (Web, SOAP/Representational State Transfer (REST), File Transfer Protocol (FTP), MOM via JMS, Mail, Transmission Control Protocol (TCP), Java Objects, etc.). Therefore the test considers both point to point and publish/subscribe models. The behavior of threads can be configured in the number of users, ramp up periods and number of test iterations. This software, like our service, also offers the possibility to user timers between each request by using mathematical distributions (Poisson, Gaussian, uniform, etc.). Besides the user can add assertions to check if the received information is correct. The source of the messages can be obtained randomly from a folder that contains files with extension .dat, .txt or .obj, but it will sequentially send the already predefined file messages. This solution provides equivalent features to our proposal except for the generation of non predefined message chains.

The IBM MQ-CPH Performance Harness was released in 2017 and it is a Java package solution for JMS service developers. This tool is used for testing the IBM middlewares WebSphere MQ and WebSphere Message Broker or other JMS servers by using a Java Naming and Directory Interface (JNDI) module. It is a command line tool that supports more than 10,000 clients [39]. It allows configuring the scenario in terms of message size, pacing, the persistence of the messages and transactionality. Different modules can, for example, wait for a message on a queue and reply to another queue (responder) or vice-versa (requestor). Despite being a powerful tool once again the scenario needs to specify the priorities of the messages (topics), size and times and although several threads with different characteristics can be configured simultaneously, without randomness it is not a real case.

Another solution proposed in the industry is called jms2009-PS [40], a benchmark based on the first industry standard SPEC-jms2007 (retired in 2016) and focused in publish/subscribe systems. One of the most interesting aspects that it offers is to configure the numbers of queues and fix the number of non-transactional/transactional and non-persistent/persistent messages. In this tool, the designers improve the filtering on the subscriber side so they can select the messages to download according to the type and location Identification (ID) of the message within a topic. To complete the design of complex traffic with which to analyze the impact on the network of users it should include timer options that are not reflected in the documentation.

In terms of most recent benchmarks, it can be found projects as OpenMessaging Benchmark [41]. It is a open source collaborative Linux Foundation effort supported by companies like Alibaba or Yahoo! that aims to provide a standard for distributed messaging based on the cloud. As the tools described above, its features include number of topics, subscribers per topic, producers and messages and the size and rate of messages. They also offer scripts to directly use on cloud platforms or analyze the flows but without any GUI available yet. Although it is a robust tool as it has the support of large companies, it does not apply to tactical networks mainly due to the possible existence of delays, memory consumption and security issues when using cloud services. On the contrary, our service can be uploaded to the cloud with the appropriate configuration and redirection.

Certain works focus on comparing two middlewares using the previous benchmarks or their own one. In [42] Chen et al. propose a Network Awareness Service (NAS) that aims to adapt applications to medium-to-long scale performance variations in mobile tactical networks. To evaluate the accuracy and reaction of their solution to a limited network where a greater workload than its capacity is loaded, they send traffic from a client. This traffic is a fixed flow that gradually increases 50 KBps every 100 seconds to exceeds the 128 Kbps link capacity. The total time of the experiment is 500 seconds. However, no specific implementation about how the traffic is injected or what is composed of thus we cannot properly compare it with our solution but we can assume that they use an unreal scenario. The research of Maheshwari presents a benchmark comparison between two middlewares, Tibco Rendezvous (TIB/RV) and Progress SonicMQ [43]. For this purpose, they create three scenarios with different scopes writing Java programs with the APIs provided by the MOMs. The first two scenarios use a fixed number of messages and publishers/subscribers with a specified message size in both point to point and publish/subscribe models. The goal of these scenarios is to analyse the average rate of messages sent and received. In the third scenario a single publisher sends messages to a specific topic. Then a subscriber compares the starting time with the time when it receives the first message to find out how quickly a subscriber can initialize the TIB/RV or Sonic broker and start receiving messages. This experiment is not reproducible as the programs used are not published and were specifically programmed to deal with RV/Sonic brokers.

Considering other tactical network studies, in [25] an adaptive middleware for Tactical Mobile Adhoc Network (MANET) applications is proposed called QoS-aware Adaptive Middleware (QAM). The authors mention that the nature of tactical networks adds a more complex of traffic than the normal one cause they involve short unpredictable and irregular bursts of data. Therefore they highlight the importance of generating random patterns of messages with different priorities that stress tactical systems. At the end of their proposal, they claim to support bursty data generation patterns as an objective of their ongoing research. On the contrary, the paper presented by Ghosh et al. [26] in the 2011 Military Communications (MILCOM) conference did not mention any model to simulate arbitrary patterns. Instead, they used a toolkit for multicast communication to send chat and images with a fixed message size and a specific flow duration. In order to prove their hierarchical queuing mechanism, Lopes et al. [1] use a synthetic message data of 500 KB to overflow the network capacity. As this work aims to improve their results, we already know that the used data flow is not real. Nevertheless, is it clear that works related to tactical networks support the need to use randomness in the tests.

In the same field of tactical networks, other studies are focusing on ad-hoc networks. This type of network is characterized by being decentralized and has dynamic topologies and proactive and reactive routing protocols. In [44] a mechanism called Dynamic Detect and Adapt Mechanism (DDAM) is introduced as a solution to monitor and adapt communications in ad-hoc networks, especially within its Agile Communications Middleware (ACM). Through four differentiated components they control and adapt the behavior of the middleware according to the network conditions. They have a GUI, a module that collects, merges and distributes the statistics received from other nodes and an Adaptive Communications Management System. In their experiments they use Mobile Ad-hoc Network Emulator (MANE)

to simulate a Tactical Edge Networks (TEN) environment. The procedure consists of varying the traffic and the type of link between two sub-networks and analysing the capacity of your solution to detect the type of link (Local Area Network (LAN), SatCom and High Frequency (HF)). Considering the emulator used we can assume that both the size of messages and the time-windows were adjustable and fixed, but we can not confirm other types of variables.

In [45], the Swiss Department of Defence proposes an application-layer routing algorithm called Tactical Ad-hoc networK Emulation (TAKE). The Swedish navy seeks to improve the robustness of its communications in MANET networks through the Optimized Link State Routing (OLSR) protocol. To do so, they develop their application focusing on routing to ensure that messages reach their destination, reducing message losses or the arrival of non-updated messages. Their software is tested both in a laboratory environment and in the field. This platform in the controlled environment allows defining the number and characteristics of the nodes that compose the topology as well as the traffic they generate. In the real environment, it allows creating profiles where it is possible to specify the size of the messages and to establish a random time-window. The priority of the messages is not essential since its objective is to make them reach the addressees.

The alternative of the Naval Research Laboratory (NRL) PROTocol Engineering Advanced Networking (PROTEAN) research group called ARL Traffic Generation Tool (ARL) [46] is a powerful tool that extends its previous proposal Multi-Generator (MGEN) [47]. ARL offers a GUI to edit the traffic flow parameters and visualize the development on a timeline. Like other previous tools it also offers analysis scripts. It is an open source software focused in IP network performance tests and measurements using TCP and UDP/IP traffic, which means that it is valid with both JMS and web services. The size of the messages is usually delimited by a minimum and a maximum so that it will vary between these two values. The priority is adjustable but it will remain fixed and there is no option to randomize. With their "burst pattern" they seek to simulate traffic such as Voice over IP (VoIP) by altering the delivery times. The pattern types available for the time windows are Poisson, uniform, periodically, burst, jitter and clone, but they do not offer Markov or other mathematical distributions yet. This tool is the most similar to our proposal.

Benchmark	**Technology**	**Message size**[1]	**Priority**[1]	**Time-Window**[1]
JBoss [34]	JMS	P/Gradually Incr.	No	No
JMeter [35]	JMS/WS	A/Fixed rate	A (topics)	Math. Distributions
IBM MQC[36]	JMS	A/Fixed rate	A (topics)	A/Fixed
jms2009PS [40]	JMS	A/Fixed rate	A (topics)	No
OpenMessaging [41]	JMS/WS	A/Fixed rate	A (topics)	A/Fixed
Chen et al. [42]	JMS	P/Gradually Incr.	A	P/Fixed
Maheshwari et al. [43]	JMS	A/Fixed rate	A (topic)	No
Ghosh et al. [25, 26]	JMS	A/Fixed rate	A	A/Fixed
Lopes et al. [1]	WS	A/Random	No	Not needed
ARL [46]	JMS/WS	A/ Random interval	A	Math Distributions
DDAM [44]	-	A/ Fixed rate	No	A/ Fixed
TAKE [45]	WS	A/ Fixed rate	No	Math Distributions
Message Benchmark	WS	A/Random	Random	Math. Distributions

[1] P = predefined, A = adjustable

Table 2.2: Benchmarks summary

Once several benchmarks have been analysed, it can be generally stated that most of them focus on the use of JMS technology as it is showed in Table 2.2. As it was declared in the Introduction chapter, the 2005 NNEC FS recommended a SOA architecture based on web services for tactical networks. It seems therefore that there is a lack of benchmarks focused on web services. Besides, the works in the literature agree on the importance of using heterogeneous traffic in experiments but do not offer realistic solutions. Our service simulates different publishers by sending messages with different priorities as if several services shared the same network and scenario. This, together with the use of mathematical distributions both in time and sequence of messages, means that a scenario close to reality can be set up to stress and test a system in tactical networks.

2.3 Final remarks

This chapter summarized the fundamental concepts related to tactical networks in order to understand the basis for the present study. Also, the most used technologies for the exchange of messages and the concept of middleware were described. Finally, other related works are described and summarized in a table in order to contextualize and compare our tool.

Chapter 3

Tactical Middleware

This chapter discusses the fundamental infrastructure around a tactical middleware which has been developed as part of the TACTICS project. As it was mentioned in the introduction chapter TACTICS is an on-going research project that addresses the communication challenges of tactical networks. The experiments in this thesis (discussed later in Chapter 5) were performed in the TACTICS testbed also using the TACTICS middleware called TSI. Thus, in this chapter we describe both the testbed and the software architecture developed within TACTICS. Then the problem that we are facing inside TACTICS and tactical networks in general is defined together with our results published in [23].

3.1 Testbed in tactical networks

The challenges experienced in a testbed using real military radios are the motivation for the development of many investigations, such as the radio buffer overflow discussed in [1]. As it was stated before, these networks are characterized by low bandwidth communication links with high delays, frequent disruptions and unpredictable mobility patterns. The Figure 3.1 illustrates a typical tactical network topology composed by three different networks connecting the nodes and the Headquarters (HQ): UHF (nominal datarate of 240 kbps, coverage of 2 km), VHF (nominal datarate of 9.6 kbps, coverage of 20 km) and SatCom (nominal datarate up to 500 kbps potentially covering the whole planet). In this network structure there are at least three types of nodes, namely: deployed, mobile and dismounted. Due to the different network coverages, it is necessary to create a backbone between the nodes at the edge and the nodes at the HQ by using VHF and SatCom.

If we consider the scalability of this simple testbed considering the radios (PR4G and Starmille) and networks used by Lopes et al. [1] the worst case scenario would have theoretically a total of $\sim$416 nodes (32 VHF radios functioning as a backbone for 14 UHF radios). This fact suggests that the user-generated data flows will often exceed the tactical network capacity. The most limiting factor is the VHF network because it has the lowest datarate (up to 9.6 kbps with PR4Gs). For that reason, the laboratory tests are focused mainly on VHF networks as it is shown in Figure 3.6. In this figure, all nodes are connected to a Virtual LAN (VLAN) switch reusing the architecture described in [48, 49].

The most challenging scenario would be the 32 expandable squads in the VHF network made up of 14 expandable nodes in the UHF network. It must be considered

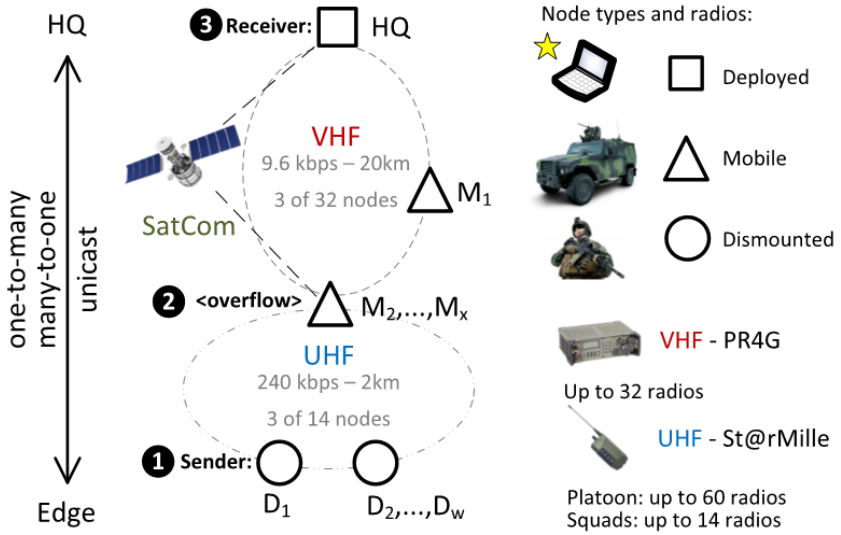


Figure 3.1: Testbed - abstract level: network topology and node types [1]

that both the nodes in the VHF and UHF networks would share the 9.6 kbps and 240 kbps respectively. At the same time, the same squad can have radios that connect two different networks. Such scenario justifies the implementation of our tool (described later in Chapter 4) because the dataflows in such a network would be difficult to predict.

The laboratory infrastructure allows testing with unicast and multicast transmission modes. Unicast transmissions are performed one-to-one, in other words in this method the data is sent from a single transmitter to a single receiver. Multicast is a method of one-to-many transmission, sending data to multiple destinations simultaneously. For example, the headquaters in Figure 3.1 could send a message to all the dismounted nodes in the edge of the network. When sending large amounts of data the multicast method saves considerably the bandwidth in the network concerning other methods such as broadcast because most of the data is sent only once. Depending on the desired method, a single receiver or several receivers are indicated and therefore the traffic in one or more nodes will be analyzed.

3.2 Architecture of TACTICS

TSI is an implementation of the reference architecture introduced during the TACTICS project that seeks to overcome the problems faced by the propagation of SOA in the tactical military environment [8]. It combines technologies relying on cross-layer information exchange to find the overlay path from source to destination through a list of proxies, delay-tolerant networks to select the best option depending on the context of the service call and store-and-forward mechanisms.

Figure 3.2 shows the TSI hierarchy of queues represented by a flow diagram. The letters indicate the two dynamic parts by labeling the change in the user dataflows with an A and the network conditions with a B. The core services of the Consultation, Command and Control (C3) taxonomy are numbered from 1 to 4 and the control points with the roman letter i to iii. In this figure, the three problems defined in the Introduction chapter (problems A, B and $A|B$) can be appreciated

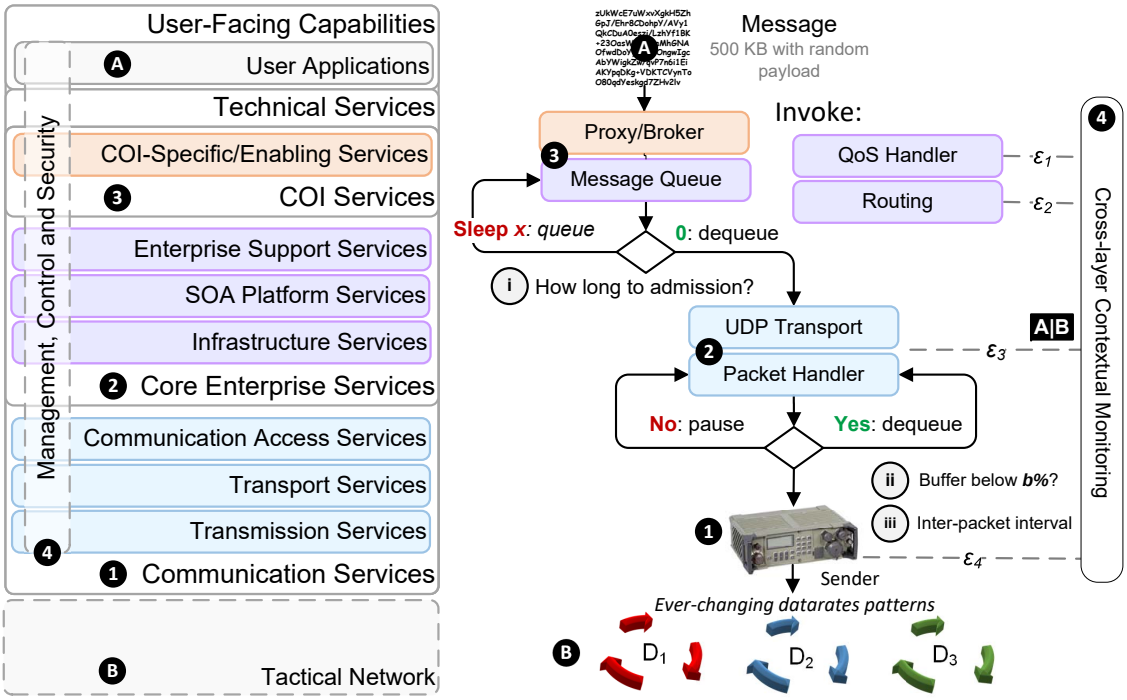


Figure 3.2: TSI queue hierarchy

again with a greater level of detail.

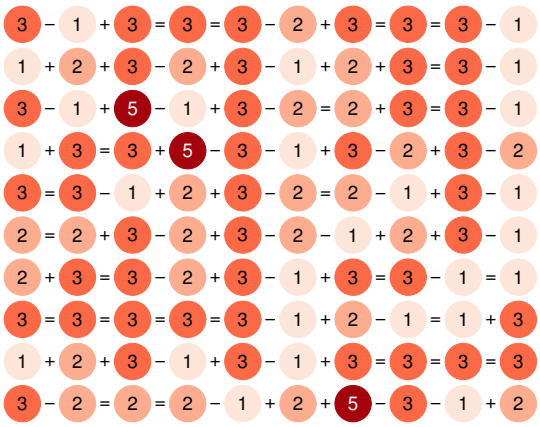
After receiving and storing the messages generated by the user services in the queue Q_m, a method called *'How long to admission?'* invoke the *QoS handler*. This method returns the expected time to finish the current data exchange. If its output is *zero* seconds, the message is dequeued and send to *UDP transport* service. Otherwise, the message queue sleeps the same amount of seconds that is given by the method. Then, the authors implemented two ways to manage the *packet handler* service that controls the IP packets. If $\Delta B_i < b\%$ being ΔB_i the radio buffer occupancy and b a predefined threshold, it sends packets. Else, it queues the packets until the buffer is below the static threshold (reactive control) or it computes the number of packets to keep the buffer usage within the b% (*proactive control*). The *proactive control* is based on introducing an IPI between packets to adjust the input data rate. Consequently, this hierarchy of queues complement each other shaping the user generated dataflow to the current network conditions also avoiding buffer overflow.

3.3 Ever-changing datarates

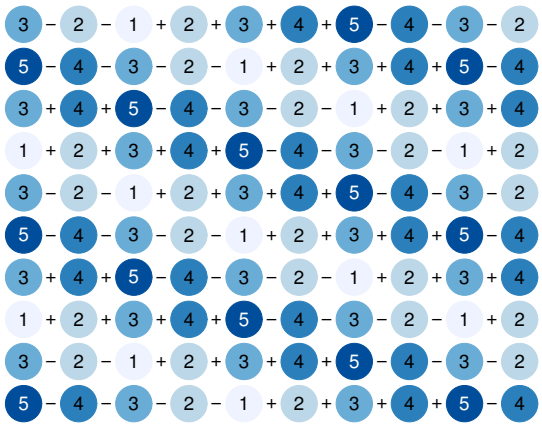
In [23], we discussed experiments by bringing together two previous investigations [1][10] to test their new queuing mechanism. In these investigations, a message four times larger than the capacity of the VHF radio buffer (indicated at the top of Figure 3.2) was used to analyze system's performance over three different patterns of datarate change, namely D_1, D_2, D_3 (shown at the bottom on Figure 3.2). Therefore the study is mainly focused on problems $A|B$ and B.

The model defined to create three different patterns alternating between the five different datarates supported by the PR4G radios (from 0.6 to 9.6 kbps) is based

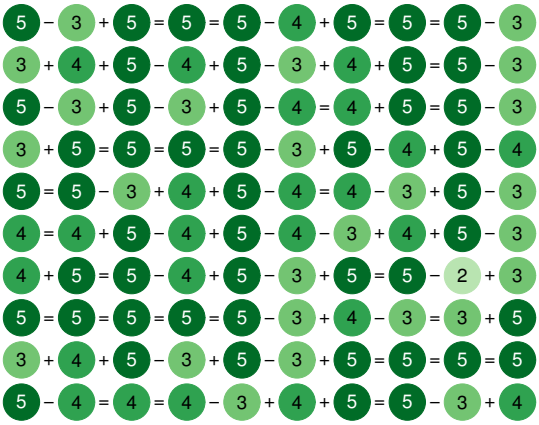
on stochastic processes (Markov chains). In Figure 3.3, the three different patterns (D_1 (red), D_2 (blue) and D_3 (green)) are shown. Notice that this figure is annotated with three symbols; '+' for datarate increment, '=' for equal datarate and '-' for datarate decrements. Each of the patterns generates a sequence of 100 states starting at the bottom left corner and ending at the top right corner. Each state is represented by a number from 1 to 5 where 1 corresponds to the lowest and 5 to the highest datarate and a lighter or darker color the smaller or larger the datarate. The same way we use it in our work, the term "ever-changing" is used because the possible combinations come from a permutation of $\binom{100}{5} = 75,287,520$ combinations. Thus the D_1 pattern represented in red color contains the lowest datarate showing a situation in which the network does not have favorable conditions. Pattern D_2 in blue illustrates a pendulum behaviour starting from the most favorable state and changing up to the most challenging. Pattern D_3 contains the highest states so it simulates a mostly propitious situation. As a result, the average datarate for the three sequences are 1.8 kbps ($\pm$ 1.57) for $D_{1\text{(red)}}$, 4.14 kbps ($\pm$ 3.12) for $D_{2\text{(blue)}}$ and 6.51 kbps ($\pm$ 3.16) for $D_{3\text{(green)}}$, therefore creating different network conditions.



(a) D_1 (red) $\sim$ 1.6 kbps

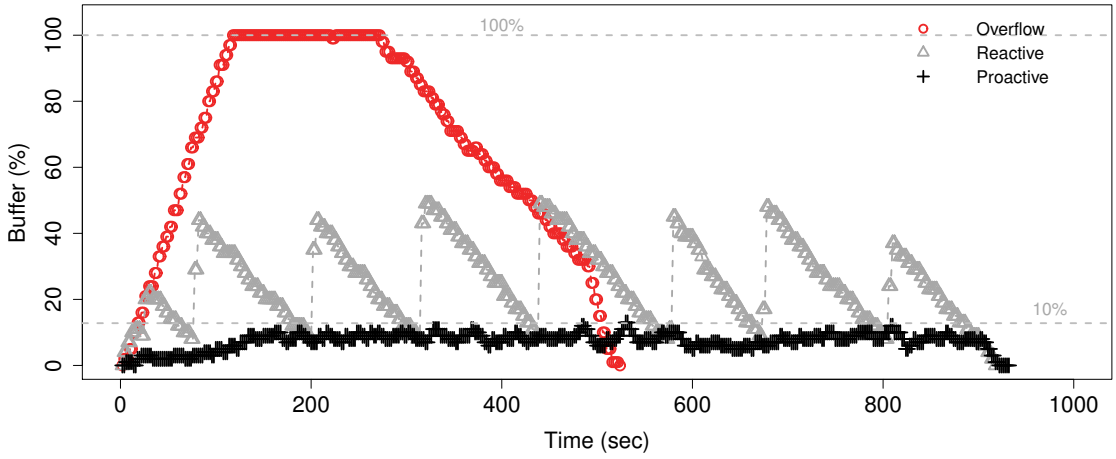


(b) D_2 (blue) $\sim$ 3.4 kbps

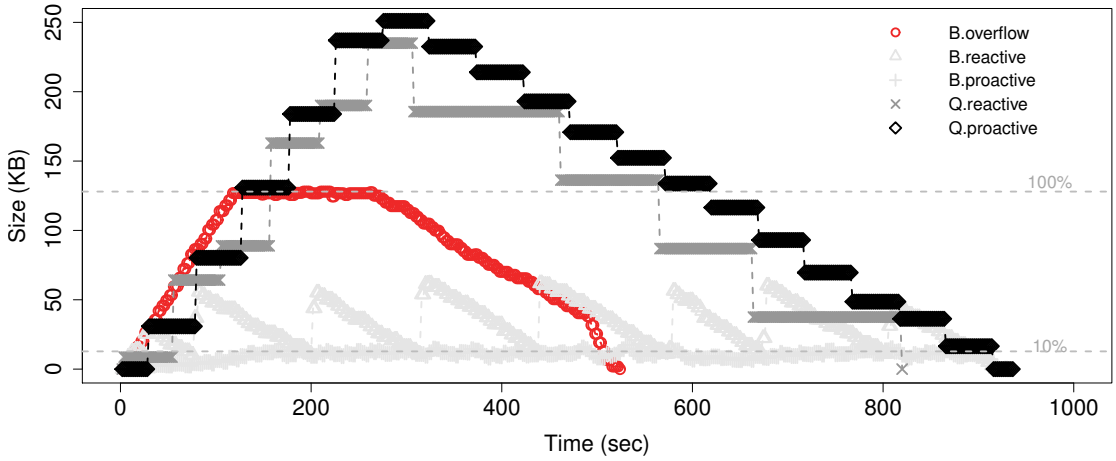


(c) D_3 (green) $\sim$ 5.4 kbps

Figure 3.3: Sequence of datarates from D_1, D_2 and D_3



(a) Radio buffer B: overflow (red), reactive (grey), proactive (black)

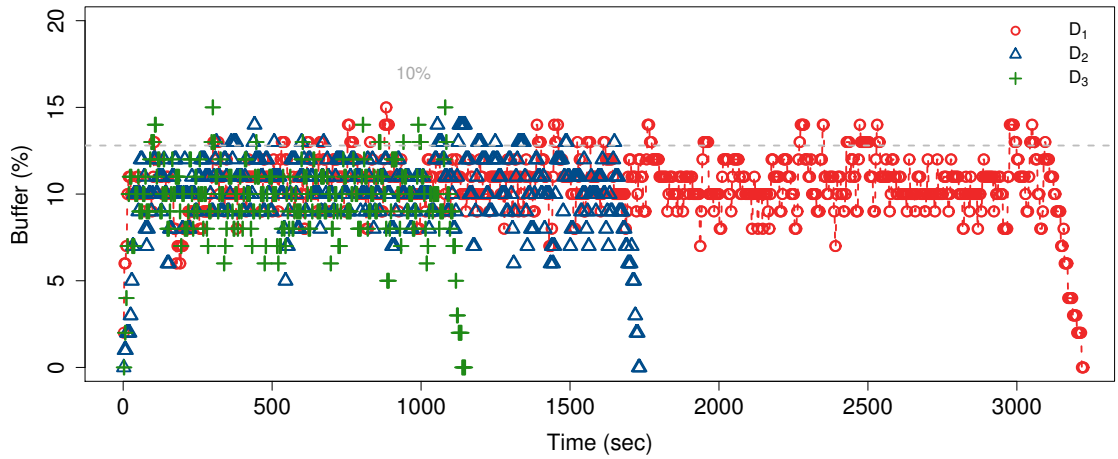


(b) Queue of packets Q: reactive (grey) and proactive (black)

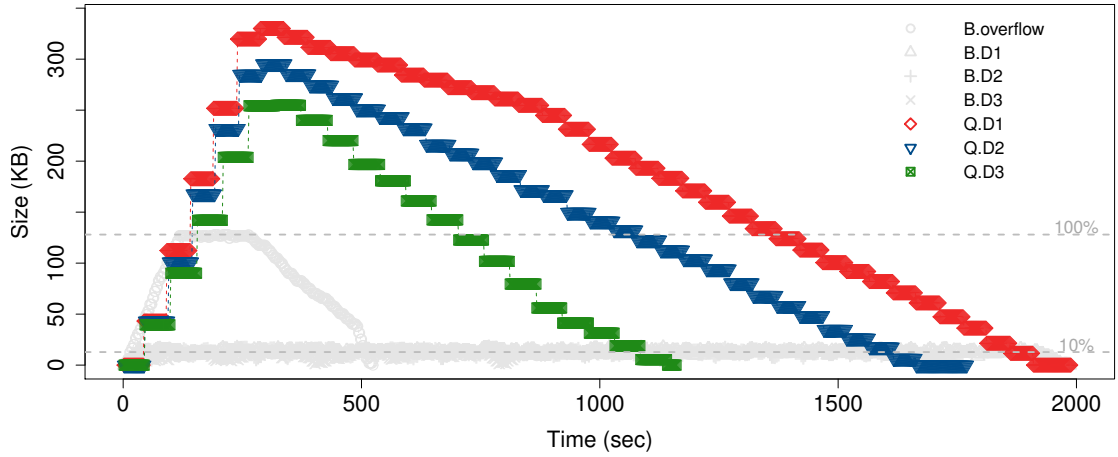
Figure 3.4: Comparison between reactive/proactive solutions

To test the performance of their hybrid solution (by combining reactive [1] and proactive [11] solutions) Lopes et al. experiment with stable laboratory conditions compared to the reactive solution. In it, they analyze the evolution of the radio buffer and queue of packets for overflow (red circles), reactive solution (grey triangles) and hybrid/proactive solution (black +). Figure 3.4a plots the buffer occupancy (%) over time, here the overflow curve is used as a baseline for the comparison to highlight that the 500 KB message overflows the radio without a mechanism to shape the dataflow. The reactive solution (grey curve) successfully sends the message creating peaks of buffer usage (up to 50% of the buffer) because the mechanism keeps sending packets until the system learns that the threshold was crossed. The proactive solution stays around the 10% predefined threshold because it uses the feedback from the radio and routing protocol. By consequence, the proactive solution (black curve) relies more on the queue of packets as plotted in Figure 3.4b. In this figure, both mechanisms have almost half of the message (∼250 KB) stored in the queue of packets after ∼5 minutes (∼300 seconds in the plot). Also notice that the size of the packets queue is almost the double than the radio buffer size (128 KB).

In sequence, they repeated the same experiment using the three patterns D_1, D_2 and D_3 to change the network conditions. The Figure 3.5a shows the evolution



(a) Radio buffer B: D_1 (red), D_2 (blue) and D_3 (green)



(b) Queue of packets Q: D_1 (red), D_2 (blue) and D_3 (green)

Figure 3.5: Radio buffer (%) and Queue of packets (KB)

of the radio buffer over time, allowing to compare the duration of each experiment. The duration is different for each of the patterns because the datarate they use is different, nevertheless, it is demonstrated that the message is sent despite the different conditions of the network. It is also observed that the occupancy of the buffer oscillates around the 10 % predefined threshold. In Figure 3.5b, queue size is plotted as a function of time showing that the hybrid solution relies more on the queue of packets during the more challenging network conditions $Q.D_{1\text{(red)}}$, followed by $Q.D_{2\text{(blue)}}$ and then by $Q.D_{3\text{(green)}}$ which is the best scenario, when compared to the other two.

3.4 Ever-changing QoS-constrained dataflows

With the previous section as a motivation to implement our solution tool to also create ever-changing QoS-constrained dataflows, we have already discussed that the forces use a variety of command and control services to communicate in tactical networks. Therefore, it is probable to have outbursts of messages in emergency scenarios like medical evacuations (Problem A). The problem here would be the bottleneck traffic created involving messages of different priorities sent at different

times and stored in a queue. To test a system that will have to deal with a situation with disconnections, delays, loss of messages, messages with different sizes and with QoS-requirements to meet it is important to have a tool to challenge the performance bounds of the hierarchy of queues developed by TACTICS.

In order to define the limits of a system in tactical networks and to be able to test and improve it, we propose a tool called message benchmark. With this we seek to fill the gap in the related literature where real situations that include the element of randomness are missing. Message benchmark is a service that simulates different scenarios by adjusting metrics as message size, message priority and time-window between messages. It offers an infinite number of possible message pattern combinations to stress the system as the user can select the desired size in KB, modify the Markov chains or probabilities and choose distributions with different means and standard deviations as discussed in details in the next chapter (Chapter 4).

3.5 Final remarks

This chapter has described the architecture of TACTICS, the problem of ever-changing datarates addressed in previous investigations and the problem of ever-changing QoS-constrained dataflows addressed by this thesis. In the purpose of simulating user dataflows that stress TACTICS store-and-forward systems, a tool has been developed. This tool is therefore located in the user application layer, at the beginning of the entire TACTICS pipeline, as shown in Figure 3.2. In this way, the limits of the middleware and the effect of the traffic injected into the network can be analyzed.

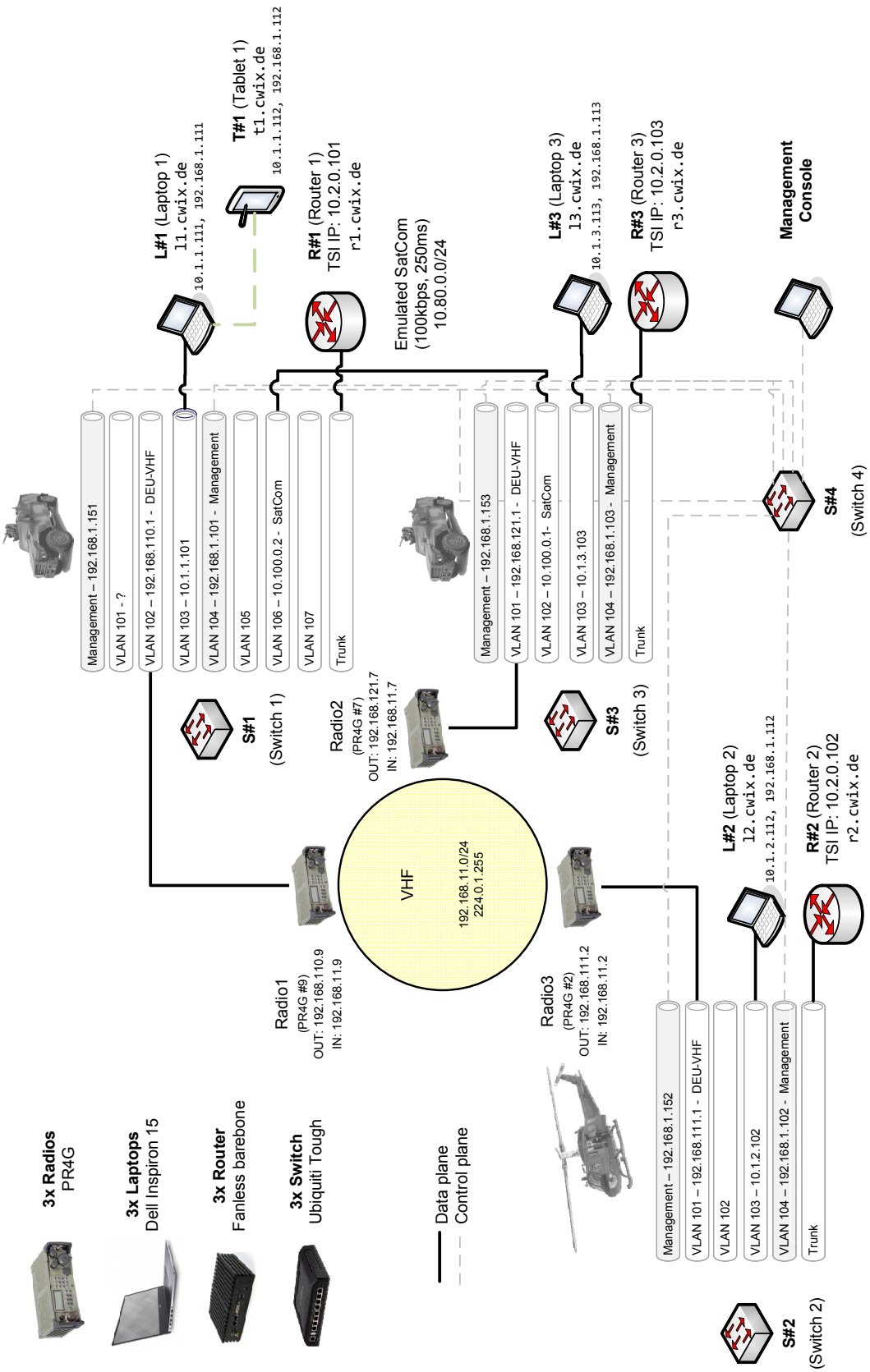


Figure 3.6: Testbed: physical network VHF

Chapter 4

Creating QoS-Constrained Dataflows

We designed a tool called "Message Benchmark" to create QoS-constrained dataflows to challenge the store-and-forward mechanisms in tactical networks. In this chapter, the methodology we used throughout this study is discussed together with a description of the resources we use to conduct the experiments. Our tool relies on the publish/subscribe infrastructure implemented by the TSI middleware. Our development reused a user service called TACTICS Obstacle Alert service (OAS), so this service is briefly explained and, finally, the implementations made in our tool to perform the experiments are described in details.

4.1 Methodology

Based on the problem statement described in Chapter 1 and assuming a set of C2 services, the methodology starts with the analysis and understanding of the TACTICS Obstacle Alert project, in order to become familiar with the infrastructure that will serve as the basis for the present work. Obstacle Alert Service (OAS) is a service used to warn of obstacles through the publish/subscribe pattern. Therefore, once understood the operation of the same the set covering the transmission and type of messages and the GUI is adapted, improved and used to the purpose of our investigatin, that is develop a tool to create different patterns of QoS-constrained dataflows.

The next step is, therefore, to decide the new functionalities to create a realistic and reproducible chain of messages simulating the user's behaviour during a particular event/mission. These, as proposed by Lopes et al. in his work [11], focus on the randomization of the priorities of the messages and we also decide to add a time-windows between them. Then, the stochastic models are developed in Java to generate the message strings. For the configuration of the priority in the messages, four different patterns were implemented: (1) stochastic model based on Markov chains, (2) model based on Markov strings where the user indicates the sequence of priorities, (3) statistical probability model and (4) a configurable default option. As for time-windows between messages, mathematical distributions were also developed (uniform, Gaussian, exponential, Poisson, log-normal and Pareto).

Finally, the flexibility of the models implemented by our tool was tested in a laboratory offered by the Fraunhofer FKIE (FKIE) as was explained in the TACTICS

chapter (Chapter 3) with different combinations of messages and QoS requirements. To do this, three experiments of a different nature (one large message, many messages in burst mode and with time-window respectively) are carried out and the impact on the network and the radio buffers is analysed.

4.1.1 Resources

Considering the setup of TACTICS, the specific part of the structure used in this thesis is composed of three PR4G Radios, three laptops, three routers and three VLAN switches; whose specifications are listed in Table 4.1. There are several VLAN connections among the switches and routers emulating/simulating different networks (e.g. SatCom, VHF, Management, etc.) that can be used one at a time since they are physically linked through an interface. Nodes are emulated using the computers and a backbone based in VHF radios is also established. The work mode consists in setting a node as manager console from which to launch the tool, a switch as manager network and the radio as the one in charge of transmitting the dataflow. Based on that, we generate the dataflows by using the message benchmark tool to stress the performance bounds of middlewares and networks.

PR4G Radio [50]	
Dimensions (WxHxD)	290x140x340 mm
RF output power	Up to 10 Watt (W)
Operating temperature	-40 ^{o}C to +70 ^{o}C
Frequency range	30-88 MHz
Data transmission	IP packet routing 4.8-38.4 Kbps
Dell Inspiron 15 [51]	
Central Processing Unit (CPU)	2.4 GHz Intel Core i7-75500U (dual-core, 4MB cache, up to 3 GHz)
Random Access Memory (RAM)	8 Gigabyte (GB) Double Data Rate (DDR) 3
Storage	1 Terabyte (TB) Hard Disk Drive (HDD) 5.400 Revolutions Per Minute (RPM)
Graphics	AMD Radeon R7 M265
Fanless Barebone Router [52]	
CPU	2 GHz Intel Celeron
RAM	DDR3 Synchronous Dynamic Random-Access Memory (SDRAM)
Voltage	12 Volt (V)
Ubiquiti Tough Switch [53]	
Processor	MIPS24K 400 MHz
System memory	64 Megabyte (MB)
Power over Ethernet (PoE) out Voltage range	22-24 VDC
Networking	Management Port 10/100 Ethernet Port
interfaces	Data Ports 10/100/1000 Ethernet Ports

Table 4.1: Specifications of the equipment composing the testbed

4.2 Publish/Subscribe messaging

The TACTICS OAS has the purpose of informing COI subscribers of the existence of an obstacle by updating the Local and Common Operational Pictures (COP) during the convoy movements. It is composed by two correlated components: (1) the service implementation itself in charge of the generation and forwarding of alert notifications and (2) the operational client where attributes of the specific scenario conditions are defined. The service implements a interface with the Notification Broker and uses a graphic interface to allows users to send the alerts to a predefined topic, thus implementing the publish/subscribe messaging pattern.

In this project we also make use of the Notification Broker core service, in particular of the Notify operation to publish the messages. To this end, the service "message benchmark" reuses the interfaces developed in the Obstacle Alert service (*NotificationProducer, RegisterPublisher and NotificationConsumer*). The reason is to take advantage of the implementation already established of the publish/subscribe message exchange in the user applications layer.

The improvements added to this service are grouped in three categories: (1) number and size of message, (2) message priority and (3) message time-window. This new service offers the possibility to generate different scenarios by setting message parameters. Notice that we are able to generate a combination of 36 different setups of messages using the features of these categories. Every new feature is defined in the following three subsections.

4.2.1 Message size

The first improvement is related to the configuration of both the number of messages to be sent and the number of sequence of messages. In order to modify the size of messages we created a sequence of random characters to change the payload of the message according to the desired magnitude indicated by the user. On the other hand, the total size of the message sequence can be also indicated or a default size can be used. In this way, the system can be tested using a specific workload defined by the user through the GUI.

4.2.2 Message priority

The priority of the message is essential to manage it in the queue of messages and give preference to the most relevant ones, depending on the strategy used. We have developed four different modes to set the message priority. In the first one, the user can choose a scenario (low, medium or high priority) based on different Markov chains described in [10], the second option generates messages following the chain of priorities configured by the user in the column "priority", described in the Application section 4.3. The third one uses a statistical probability model based on the new column "probability" in the GUI also explained in the Application section 4.3 and the last one uses priorities by default.

A Markov chain is a specific stochastic process in the field of probabilistic models [54]. Discrete-time Markov chains are those in which the state changes at a certain discrete time. The state of the chain in the instant t is denoted by x_t and it belongs to a specific set of possible states. As the time goes by, changes of state take place in probabilistic terms and are represented through of the so-called probabilities of transition between states, which in the case of transitions in a stage corresponds to the probability of moving from a state to another from a time step t to the next t + 1.

4.2.3 Message time-window

The time-window is the time that the system waits between sending each sequence of message. In this context, other probabilistic models can be used to simulate the human behavior with the element of randomness. For example, Bernoulli and Poisson

distributions that use a discretization of the time into periods or a continuous-time respectively. One is based on an arrival p per trial and the other on an arrival rate λ per unit of time [55]. For that reason we decided to offer two options, static time window or dynamic (choosing a distribution). The dynamic time-window is generated using the configuration chosen by the user in one of the following distributions explained in the next section 4.3: uniform, Gaussian, exponential, Poisson, log-normal or Pareto.

4.3 The Message Benchmark Application

The interaction with the user is done through a GUI in order to simplify the modification of the variables to be used in the realization of the experiments, as shown in Figure 4.1. Next, we match each function (numbers from 1 to 8 in the figure) and its respective explanation as following:

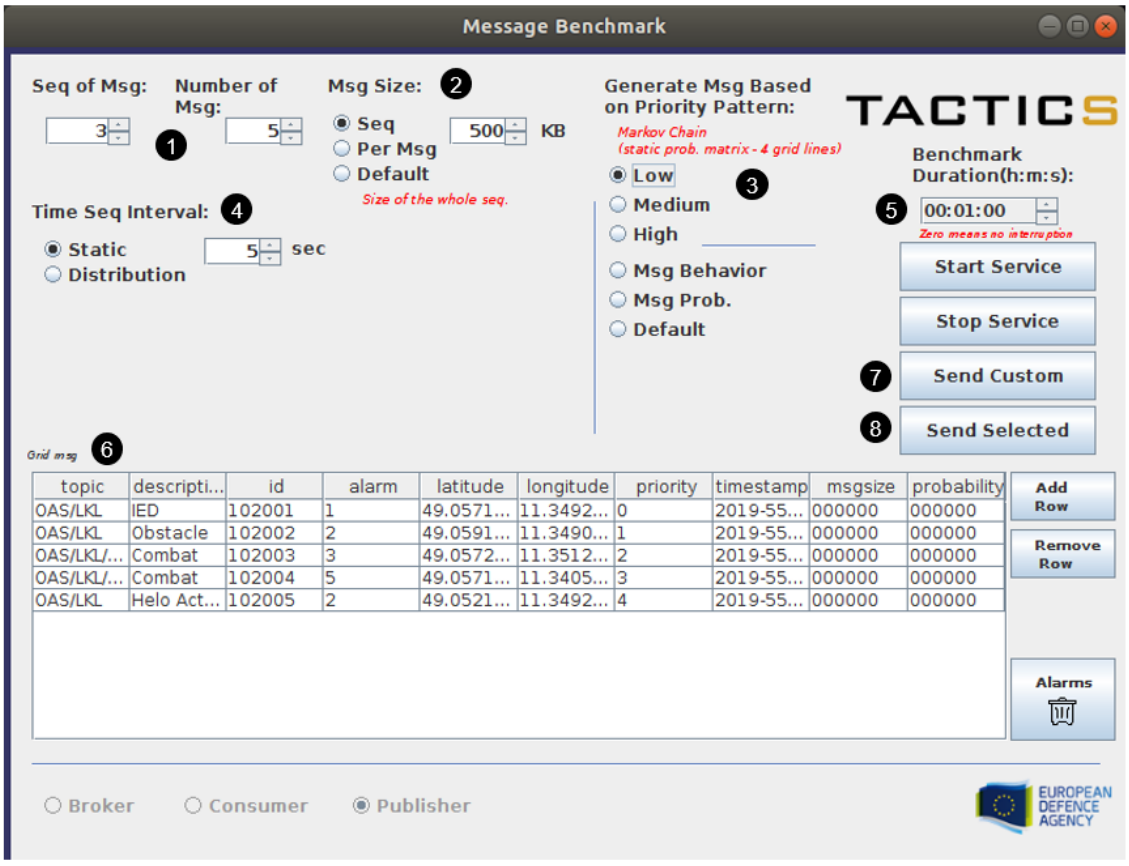


Figure 4.1: Graphical User Interface of Message Benchmark tool

1. **Sequence**: first it is needed to specify the number of messages per sequence and the message sequences to send. Considering a sequence as a set of messages composed by different messages. These parameters are used to stress the system and simulate different user behaviors by adding workload to the network.

2. **Size**: in the second step the size of the sequence or messages individually is defined. Using one of the three available buttons the user can (a) indicate the

desired sequence size in KB, which automatically will calculate the size per message, (b) indicate the desired size per message individually or (c) choose the default size messages. To achieve the requested message size the algorithm adjusts the length of the description field (see column description in Figure 4.1).

3. **Priority**: there are four different options for defining the priority of messages. By selecting the "low", "medium" or "high" buttons, the user indicates which scenario he wants to use according to the Markov chain used. In this way, the *high* scenario would be the one with the highest probabilities of change of state (service) and the *low* the one with the least, simulating a more chaotic situation or less. The election of *"Msg. Behavior"* allows the user to specify the data with priorities in the "priority" column, as for example using the information about previous experiences. Our service will generate a chain of messages employing a Markov probability matrix based on the priority preferences that the user has previously indicated. On the other hand, the button *"Msg prob."* uses a model based on the probabilities of the *"probability"* column and finally, the *"default"* button fires the algorithm using the messages offered by default.

4. **Time sequence**: the last configuration step defines the time-window mode. By selecting the static one the time-window will remain fixed until the end of the experiment. The dynamic option allows choosing one of the distributions indicating previously the mean, standard deviation or interval according to the model. As a result, the algorithm will generate different time-windows that will be embedded between the message sequences.

5. **Duration**: in the upper right corner of the GUI there is a counter to define if a time of duration of the experiment is desired. Using this parameter it is possible to observe for example how the middleware treats a string of messages that has been interrupted.

6. **Messages**: at the bottom of the GUI there is a table in which the user can modify any of the messages as desired. Adding or deleting rows and modifying the values in the columns. If the user wants to use, for instance, the *"Msg. behaviour"* priority option the user could insert and/or modify new rows with the data he wants to analyze.

7. **"Send Custom" button**: this button launches the application including all the messages presented in the table and taking into account the parameters indicated by the user.

8. **"Send Selected" button**: this button is in charge of launching the application considering the parameters indicated by the user and the message selected in the table.

The service message benchmark allows configuring the setup according to the user's needs. In addition to the description of our application, the taken flow by the GUI can be seen in Figure 4.2, following the configuration process from left to right and from top to bottom.

To exemplify the use of *Message Benchmark* tool, as show in Figure 4.1, the user created a scenario with 3 sequence of messages with 5 message each and size of 500

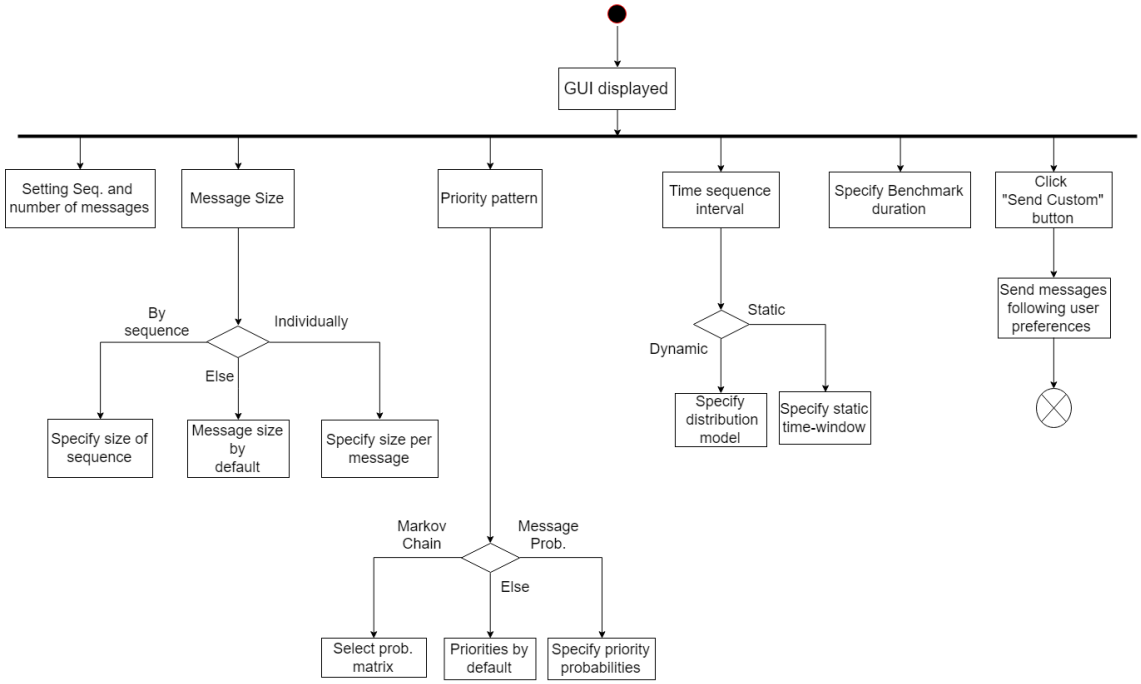


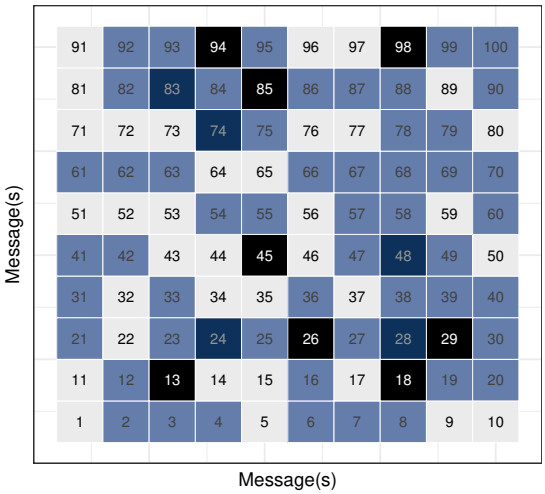
Figure 4.2: UML activity diagram

KB. The priority of the five messages is randomly generated following a probability matrix (Markov) in the *low* option (stable scenario) and the time-window between each sequence set to five seconds. The total duration of the experiment is one minute and the message parameters can be seen in the table at the bottom (notice that the algorithm will change the size of the description to reach the desired message size).

The general structure of classes composing the message benchmark project is shown in the UML class diagram in Figure 4.8. The main class of the project is *obstacleAlertGUI* because it is in charge of controlling and launching the service. The most relevant classes are explained below.

The class *Tactics* was modified to support new features of the message benchmark. When the user calls *"Send Custom"* button the array obtained from the GUI is stored by the algorithm 1 (Appendix C) as a *LinkedHashMap* and using the publish/subscribe protocol the messages will be sent to the system pipeline with different priorities depending on the topic selected. This means that a message with the same content can acquire a different priority in the message queue depending on which topic it belongs to. It is therefore necessary to distinguish between the priorities that some topics have in front of others and at the same time some messages in front of others within a specific topic. The array that receives *Tactics* from the GUI contains the position of the messages concerning the table to be sent sequentially already ordered. In this way, this class only focuses on sending the requested messages.

The class *MarkovMsgGeneratorProbMatrix* generates sequences of messages following the model introduced in [11]. First, a matrix A represents the Markov chain of the four services (s_1,s_2,s_3,s_4) and the relation between them (i.e. the conditional probabilities). The services are previously defined in Table 1.1. Then, the Algorithm 2 (Appendix C) will create a sequence of a desired number of messages numbered as shown in Figure 4.3 from left to right indicating the progression in which it will



(a) Sequence of messages.

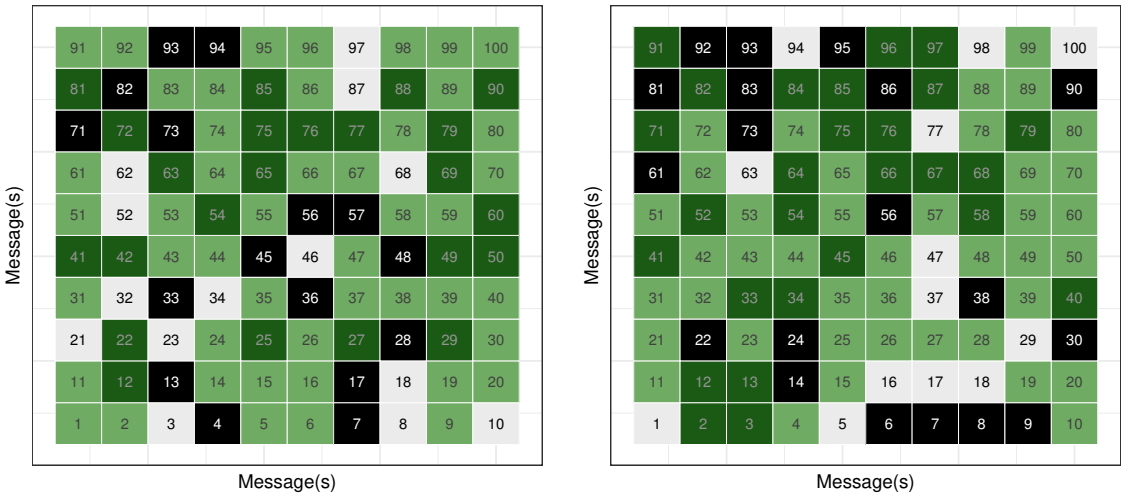
Figure 4.3: Sequences of Messages

be sent to the tactical network. The darker the color of the ordered message, the higher priority it will have in the message queue: *0 flash* (black), *1 immediate*, *2 priority* and *3 routine* (gray). We have associated blue color with the low scenario, green with medium and red with high. The Results chapter shows a visual comparison of the behavior of the different options to associate a priority to the messages generated by our tool.

To show the randomness of the developed model it is enough to execute it several times. As a consequence, different message strings are generated. As can be seen in Figure 4.4, using different seeds, sequences of 100 messages with different compositions are obtained. Since Figures 4.4a, 4.4b, 4.4c and 4.4d has 16, 19, 15 and 16 high priority messages randomly distributed respectively. By observing the color pattern it is also possible to easily discern the differences in number and allocation of each type of message.

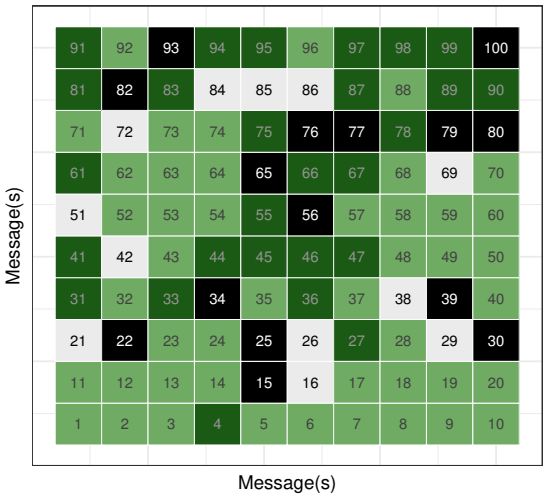
For a better understanding and testing of the model, an alternative script has also been developed. It generates and presents smaller sequences (20 messages) with 20 queues side-by-side. This allows to measure the effectiveness of the algorithm when generating strings of messages with different nature. Figure 4.5 shows the result of using three different matrices (A_1, A_2 and A_3) representing three different scenarios differentiated by colors as discussed above. Figures 4.5a, 4.5b, 4.5c, 4.5d, 4.5e and 4.5f show the different strings in terms of message priority and order of arrival.

The script was firstly developed in R [56] and then transformed to Java in order to be able to upload the code to the network using the open source software Ansible [57]. As a result, the future chain of messages is stored in an array indicating its order of arrival, priority and position in the sequence. Different Markov chains are used to simulate different scenarios, from more stable scenarios to scenarios that are more likely to change between states as can be seen in the simulation matrices defined in the result section. The distribution model that forms the basis of the algorithm is a sequence of random variables X_1, X_2, X_3, ... X_t where the probability of moving to the next state depends only on the present state and not on the previous states. Being s_t the state of the process at time t, t = 0,1,2... as follows:

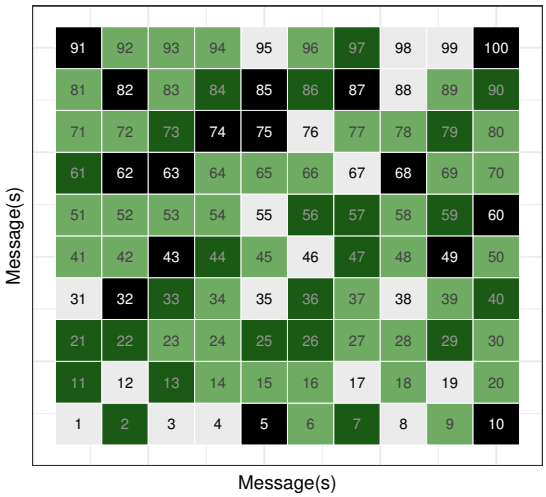


(a) First run

(b) Second run



(c) Third run



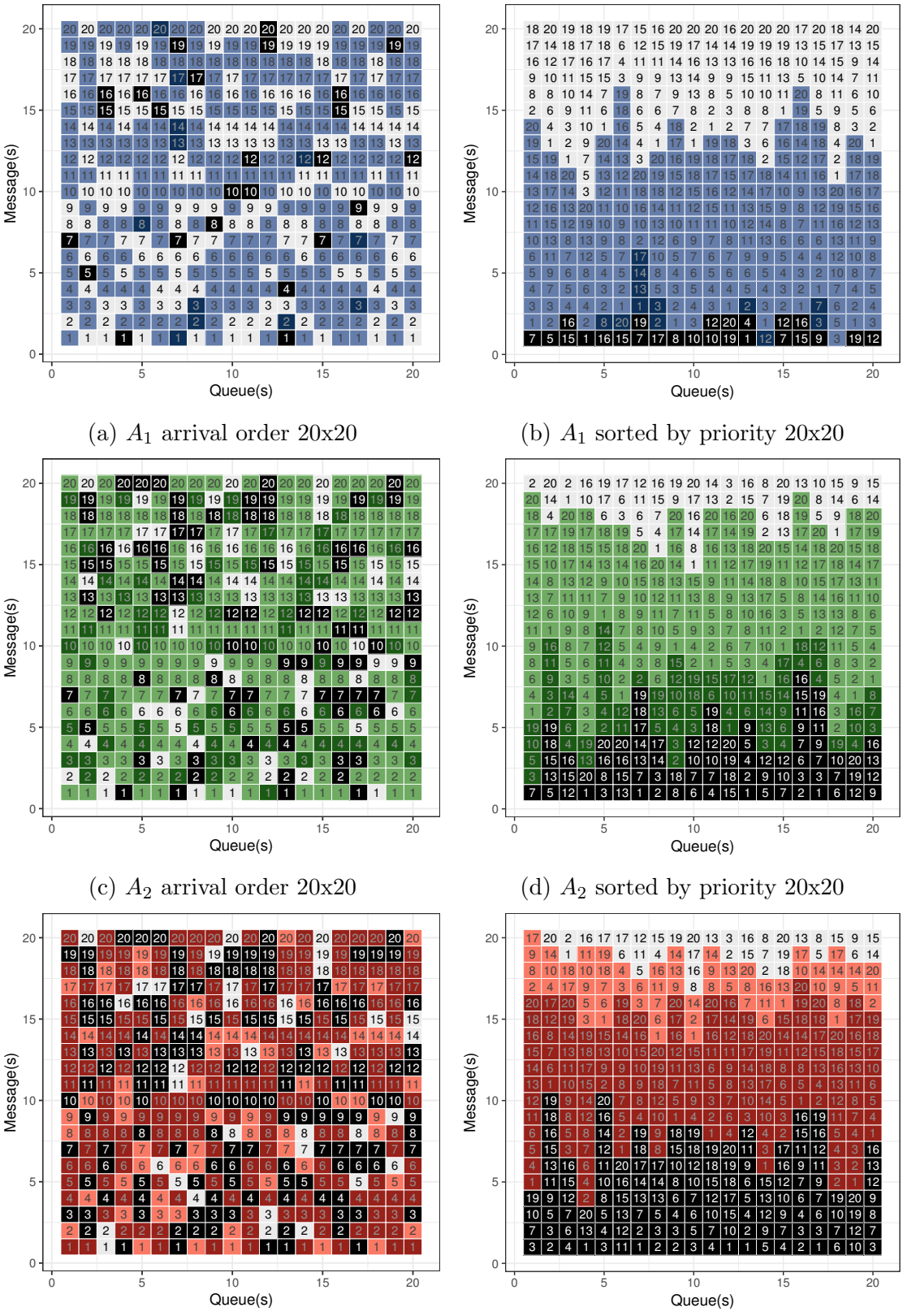
(d) Fourth run

Figure 4.4: Sequences of Messages using different seeds

$$P(X_{t+1} = s_{t+1} | X_t = s_t, X_{t-1} = s_{t-1}, \ldots, X_0 = s_0) = P(X_{t+1} = s_{t+1} | X_t = s_t) \quad (4.1)$$

The transition probability matrix P in 4.2 represents the probabilities of moving from i to j in one time step. Considering i as the row and j as the column and always changing the state in the finite state space S, for example, $P_{3,2}$ is the probability of changing from state 3 to state 2. In Figure 4.6 a generic state diagram can be observed.

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & P_{3,2} & P_{3,3} & P_{3,3} \\ P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} \end{pmatrix} \end{array} \quad (4.2)$$



(a) A_1 arrival order 20x20

(b) A_1 sorted by priority 20x20

(c) A_2 arrival order 20x20

(d) A_2 sorted by priority 20x20

(e) A_3 arrival order 20x20

(f) A_3 sorted by priority 20x20

Figure 4.5: Twenty queues with 20 messages from A_1, A_2 and A_3

The class *MarkovMsgGeneratorInputPattern* takes the user priority sequence in the column "priority" to generates a probability matrix and after creating an array of

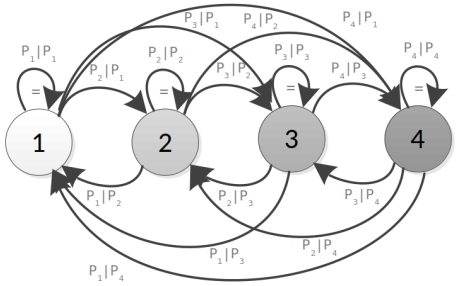
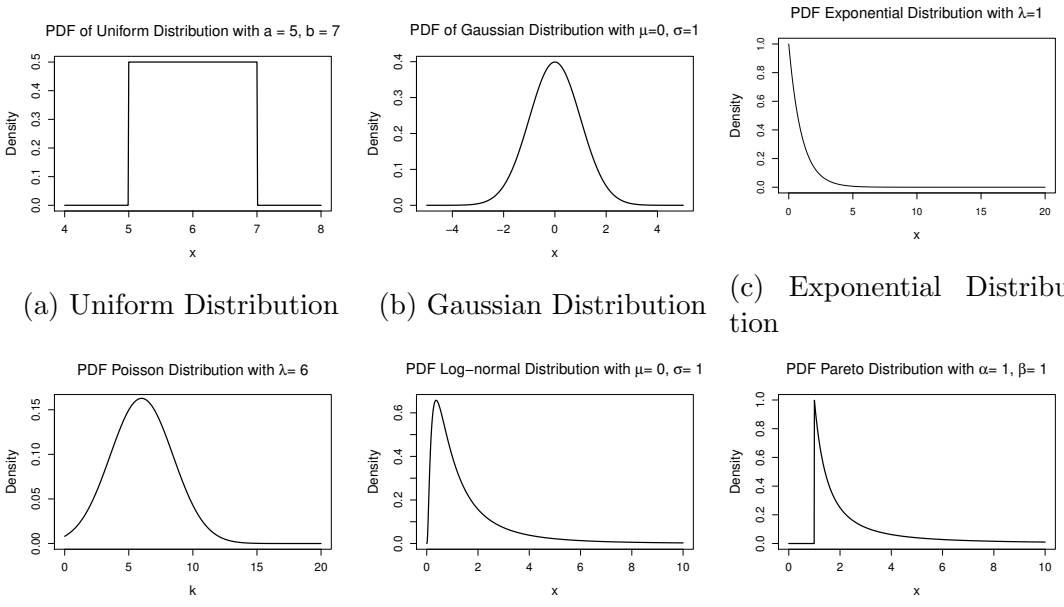


Figure 4.6: Generic Markov chain

messages following the Markov chain model. This option can be seen as the opposite of the previous Markov function since in this case, the user gives the sequence of priorities messages and the algorithm generates an array from it. While in the first Markov option the user needs to define the probability matrices in order to obtain an array from it. Therefore, a use case for this option could be the use of data from past tactical exercises to experiment with them in another context.

Similarly, the class *ProbSequenceOfMsgGenerator* generates an array with the messages to send according to the probabilities indicated in the GUI table. So the user can create a scenario where the messages he wants are more likely to be generated according to their needs, thus simulating the greater or lesser use of one service or another in the tactical field. Or, on the contrary, a case in which all services are used equally. The operation of the algorithm is based on a discrete distribution of probabilities. Let consider a discrete random variable X and u_0, u_1,... i be the values it can take (message type), the associated formula is therefore:

$$\sum_i P(X = u_i) = 1 \tag{4.3}$$



(a) Uniform Distribution (b) Gaussian Distribution (c) Exponential Distribution

(d) Poisson Distribution (e) Log-normal Distribution (f) Pareto Distribution

Figure 4.7: Time distributions

Finally, the different mathematical distributions are generated by the class *RandomNumberGenerator* to use them as time-window intervals. The generated arrays contain a sequence of numbers (in seconds) that behaves as a time-window between each sequence of message sent. The available distributions are defined as follows:

- Uniform distribution where a and b are the minimum and maximum boundaries and the user can configure their values. Figure 4.7a shows a possible user setup.

$$\text{Uniform distribution:} f(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b, \\ 0 & x < a \ or \ x > b. \end{cases} \tag{4.4}$$

- Gaussian distribution where μ is the mean, σ the standard deviation and σ^2 the variance. The user can modify the values of both mean and standard deviation as for example in Figure 4.7b, with a mean of 0 and an standard deviation of 1.

$$\text{Gaussian distribution:} f(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{4.5}$$

- Exponential distribution where λ is the rate parameter whose value can be changed in the GUI. Figure 4.7c represents an exponential distribution with λ 1.

$$\text{Exponential distribution:} f(k;\lambda) = \begin{cases} \frac{e^{-\lambda}\lambda^k}{k!} & x \geq 0, \\ 0 & x < 0. \end{cases} \tag{4.6}$$

- Poisson distribution where k is the number of times an event occurs in an interval and the average number of events in an interval is λ. The mean can be adjusted, Figure 4.7d uses a distribution with a mean of 6.

$$\text{Poisson distribution: } f(k;\lambda) = \frac{e^{-\lambda}\lambda^k}{k!} \tag{4.7}$$

- Log normal distribution where μ and σ are the mean and standard deviation of the logarithm. By default our tool offers a distribution with mean and standard deviation of values 0 and 1 respectively as it can be seen in Figure 4.7e.

$$\text{Log-normal distribution:} f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-(\log(x)-\mu)^2/2\sigma^2} \tag{4.8}$$

- Pareto distribution where α and β are the scale and shape parameters. By default our tool offers a distribution with scale and shape of value 1 as it can be seen in Figure 4.7f.

$$\text{Pareto distribution:} f(x|\alpha, \beta) = \begin{cases} \frac{\beta\alpha^\beta}{x^{\beta+1}} & \text{if } \alpha \leq x < \infty \ \ \alpha, \beta > 0 \\ 0 & \text{else} \end{cases} \tag{4.9}$$

4.4 Final remarks

In this chapter, we summarized the methodology we used to design the Message Benchmark tool as well as its functionality and the environment used to test it. Our tool allows establishing parameters in terms of the number of sequences, number, and size of messages, priority pattern and time-window pattern. In order to evaluate the use of it, in Chapter 5, we defined three use cases, out of 36 possible setups, to show the results we are able to obtain in a VHF network.

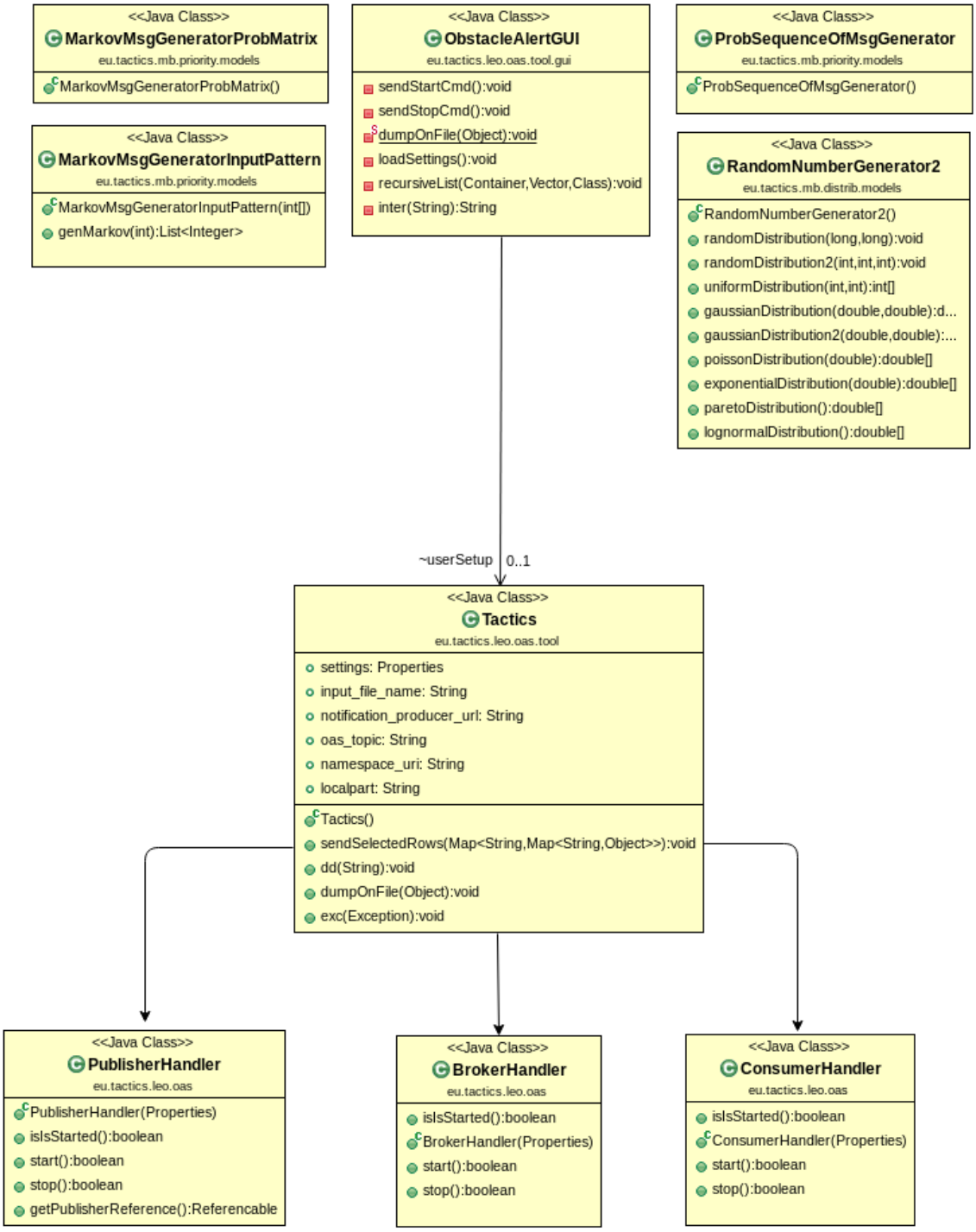


Figure 4.8: UML class diagram for the Message Benchmark tool

Chapter 5

Experiments and Results

This research explores different ways of generating message sequences with QoS restrictions to test their impact on tactical middlewares. To achieve it, two types of experiments are carried out, with and without a tactical middleware (i.e. the TSI implemented during the TACTICS project described in Chapter 3. Both analysis use tree different combinations of our tool based on the following experiments: (1) sending just one large message, (2) a burst of messages without a predefined time-window, and (3) a burst of messages with time-window distribution. Notice that our tool is able to generate a combination of 36 different setups of messages using the features described in Chapter 4, and also our tool is generic enough to be used in different types of networks. However, we choose three combinations to show that we achieved the goal of stress the network and the store-and-forward mechanism within the TSI tactical middleware. Complementing, we performed simulations creating sequences of QoS-constrained messages using the priority patterns mentioned in Chapter 4. The goal was to check its correct functioning before sending them to the tactical network.

5.1 Experiments definition

The experiments were done in two different scenarios, with and without the use of the tactical middleware TSI. First, we fixed the radio datarate at 9.6 Kbps in both scenarios. Then, in the first scenario, without middleware, the system does not have the packet handler service (described in Chapter 3) and it is a simple case where a source sends a message through the publish/subscribe model implemented by a notification broker (unicast mode). The goal of this first case is to evaluate the behavior of the system and identify the set of combinations able to stress the system in order to break it (i.e. overflow the radio buffer). On the other hand, the middleware scenario seeks to test the performance bounds of the TSI.

Three different experiments are carried out in this section differentiated with the colors *red*, *green* and *blue* respectively, the configuration setups and graphs are described below and summarized in Table 5.1.

- **Experiment 1 (red):** For the first experiment, our tool generates only one message. Therefore we choose a sequence of one message of 1000 KB. The priority mode selected is by default because in this case there are not several messages that have to compete in the message queue. The time-window between messages is set to zero seconds.

Experiment	Sequences	Messages	Size	Priority	Time-window
1	1	1	1000 KB	By default	0 secs
2	1000	1	1 KB	By default	0 secs
3	1000	1	1 KB	By default	Exp - mean 5 s

Table 5.1: Set up for the three experiments

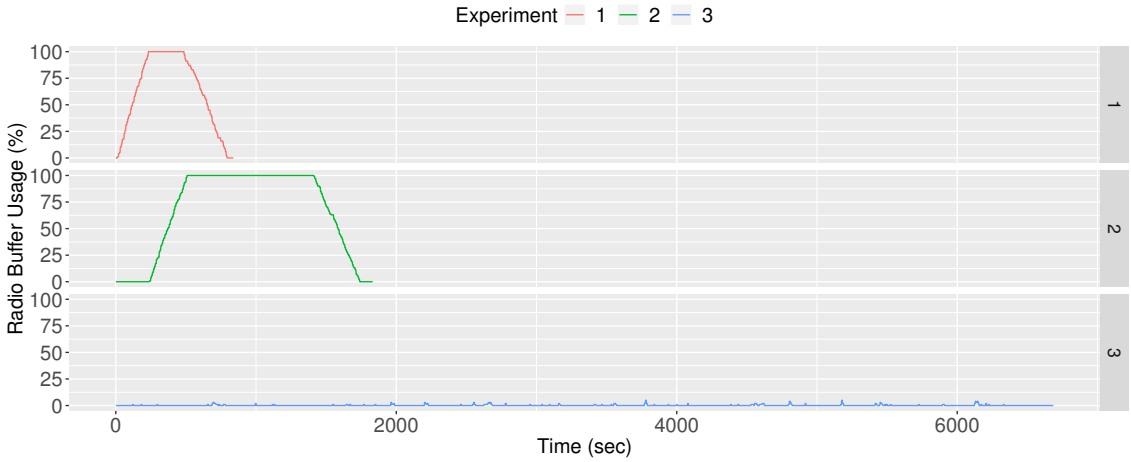
- **Experiment 2 (green):** For the second experiment, our tool generates 1000 messages. Therefore, we choose one thousand sequences of one message of 1 KB. The priority mode selected is also by default. The time-window is set to zero seconds to simulate a burst of messages, this case could emulate situations as talk outburst which might result from VoIP applications [58]. With the course of a conversation the activity of the user can be random, when a burst occurs depending on the compression codec used, messages can be generated periodically with different speeds and sizes depending on the type used.

- **Experiment 3 (blue):** In the third experiment, our tool also generates 1000 messages. The configuration is the same as the previous experiment except for the use of a time-window based on an exponential distribution of mean five. This experiment simulates a more favorable case for the network compared to the previous two since it is not a burst mode.

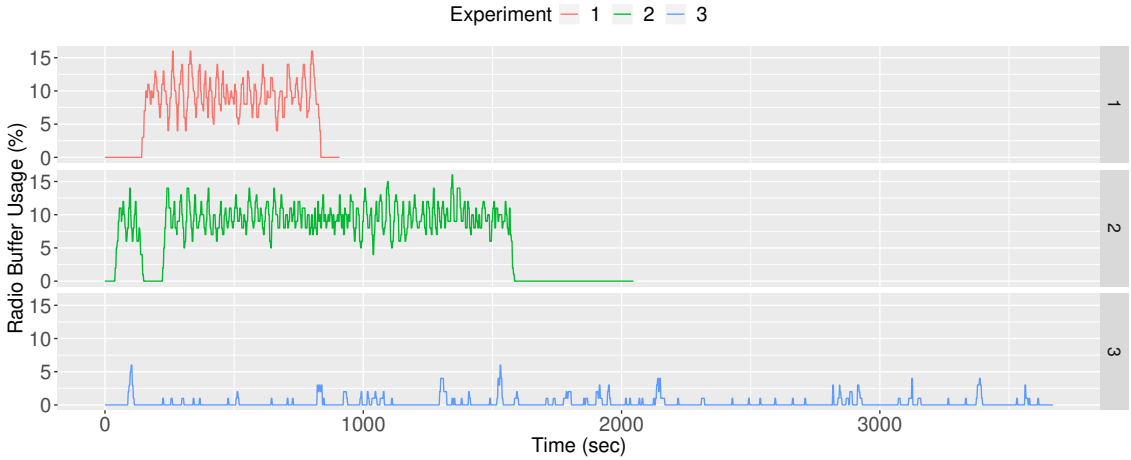
The generation of scenarios with different types of sequences allows the accomplishment of the following tasks. Firstly, we will be able to evaluate the evolution of the radio buffer as a function of time to check the correct operation of our tool in terms of sequence and number of messages. In addition to the accuracy in the handling of time-windows. Secondly, this helps to ascertain the possible loss of packets and information through the use of the UDP transport service implemented by the publish/subscribe broker and the tactical middleware. Thirdly, this serve to compare the use or absence of middlewares networks and the performance of the TSI middleware.

5.2 Experimental Results

In this section we present and discuss the quantitative the results of our set of experiments. The graphs shown in Figure 5.1 are the results of the radio buffer occupancy as a function of time and the comparison between the number of IP packets together with the delay between IP packets sent and received . The radio buffer occupancy is gathered through a Java servlet that monitors its status according to a refreshing time, in the case of the experiments reported here it was set to 2 second. The IPI represents the time between IP packets that were developed in [1] to shape the data flow according to the current network conditions. If the conditions are good (e.g. there is a high datarate) the IPI will be small and vice-verse. Figure 5.1 plots both scenarios without (a) and with the tactical middleware (b).



(a) Radio buffer without middleware



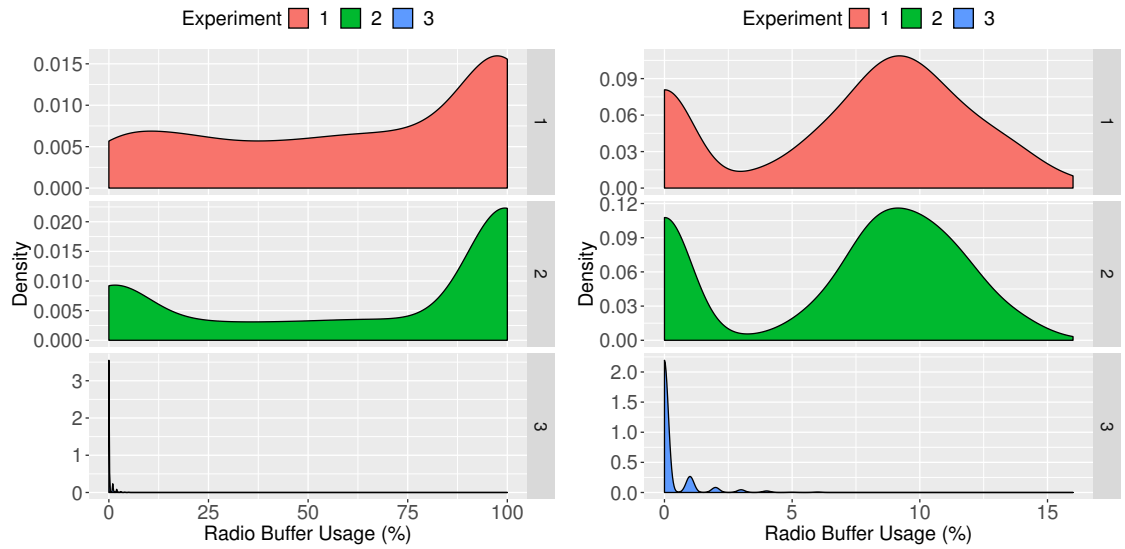
(b) Radio buffer with TSI

Figure 5.1: Radio buffer over time

In the first scenario, without middleware in Figure 5.1a, both experiments 1 and 2, the radio buffer reaches its maximum capacity giving rise to overflow as can be observed in the plot. Therefore it can be said that when sending large traffic in "burst" mode the radio is stressed. On the other hand, when a time-window is set, the radio has enough time to store and transmit the messages. It can also be observed the duration of each of the three experiments, the last being the one that lasts the longest using a time-window between messages following the exponential distribution.

In the second scenario, Figure 5.1b, the buffer occupancy stays below 15% due to the store-and-forward mechanisms within TSI. The operation of this algorithm causes an oscillation around the 10% in the first two experiments while in the third one it is not necessary, due to the radio itself can deal with that amount of information (i.e. the QoS-constrained dataflow was creating a datarate within the network capacity). The threshold of 10% is not a fortuitous occurrence, since in [23] it was decided to set that percentage.

The Figure 5.2a shows that due to the fastest possible sending of messages, the highest density is accumulated in values close to 100% until reach the total buffer capacity in the case of the first two experiments while in the third experiment the



(a) Radio buffer density without middleware (b) Radio buffer density with TSI

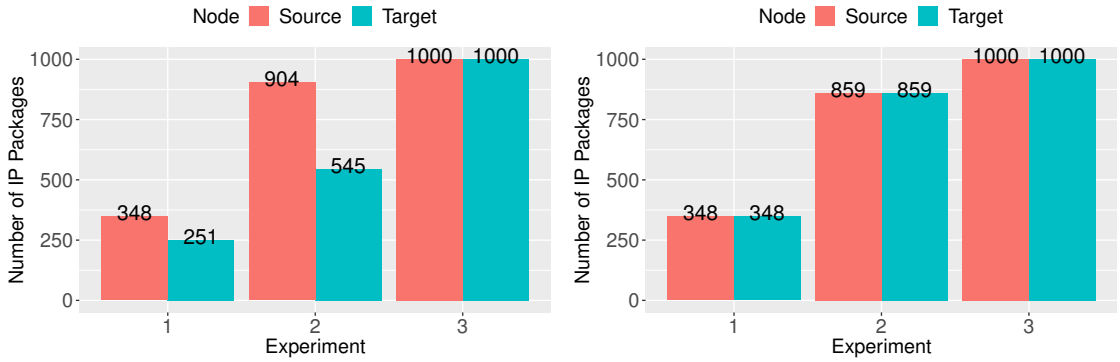
Figure 5.2: Radio buffer density

density is centered around 0% of buffer usability. This is due to the fact that the radio has to deal with a large amount of information in a short period overflowing while in the third we see the buffer underflow. On the other hand, Figure 5.2b shows a density around 9% and another peak in 0% for experiments 1 and 2. The density of buffer usage in experiment 3 maintain around 0%, showing also the buffer underflow. It is also interesting to mention the Gaussian distribution around the threshold for experiments 1 and 2 of scenario two.

It is necessary to consider the differences between the two graphs in Figure 5.1 in relation to the dimensions in the percentage and density axis of the radius buffer used as they cannot be directly compared. Comparing both scenarios it can be seen that the buffer control algorithms in TSI work correctly in unicast mode avoiding radio overflow. As a consequence we can claim that no information is lost during communication with data sets of up to 1000 KB (experiment 1) and that it can probably handle longer sequences. This is essential when dealing with situations where bottleneck might occur. It has also been verified that the variables related to the size and time-window of our tool work as expected. Demonstrating that the smaller the time-windows the higher the stress for the store-and-forward mechanism.

The first scenario presents IP packet loss during transmission in experiments 1 and 2 as plotted in Figure 5.3. This fact confirms the above mentioned concerning radio overflow as the number of IP packets sent does not coincide with those received in the first two experiments. Figure 5.3a illustrates the higher proportion of packets lost when burst mode involves a large number of small messages versus a large message. The fragmentation carried out in the transport layer is different according to the type of traffic, the 1000 KB message is divided into 384 packets (experiment 1) while the 1000 messages of the second experiment are divided into about 900 packets. On the other hand, in experiment 3, which does not use burst mode, each 1KB message fits into a single UDP packet, so we could see 1000 1KB messages.

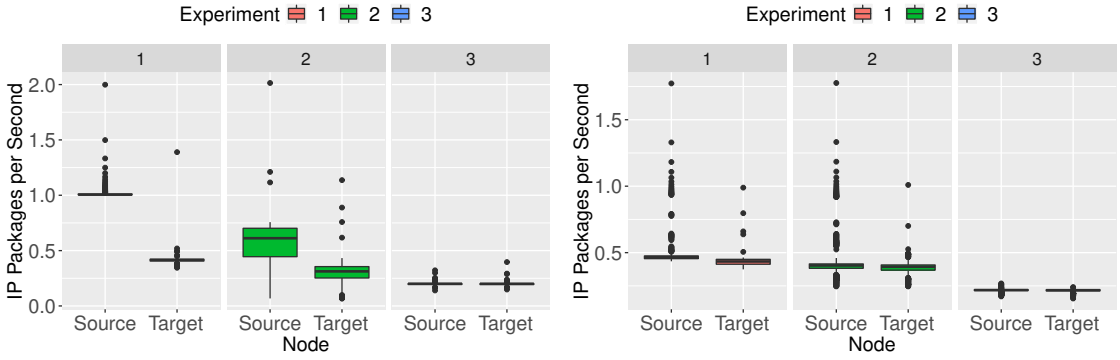
The second scenario, like the first one, presents a curious fact in the number of packages delivered and received, see Figure 5.3b. All experiments sent and re-



(a) IP packets without middleware (b) IP packets with TSI

Figure 5.3: IP packets sent and received

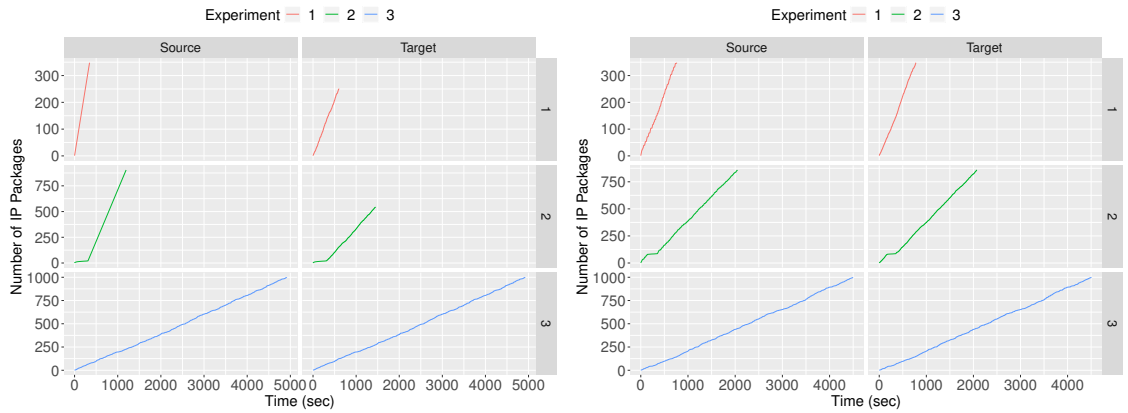
ceived the same number of IP packets as expected since the buffer is not flooded. But experiment 2 highlights the fact of fragmenting the 1000 messages into a lower number of packets. This is due to the use of GZIP, this open source software compresses all messages in the message queue before splitting into packets. It is based on the Deflate algorithm, which is a combination of LZ77 and Huffman encoding [59]. Therefore there are more compression options in large messages than in small ones and therefore more possibilities of assembling several into one single fragment. As experiment 3 offers enough time to send messages without saturating the message queue the result is 1000 messages while in burst mode (experiment 1 and 2) the number of packets is less than the number of messages.



(a) IP packets without middleware (b) IP packets with TSI

Figure 5.4: IP packets sent and received per second

The graphs in Figure 5.4 shows the rate of packets sent per second. Since they are evaluated in the same period of time, the sending and receiving rates will be significantly different in case of communication losses. Moreover, the compression also reduces the number of IP packets send per second. We can see in Figure 5.4a (in experiment 1 and 2) a gap between sources and targets, as a result of a buffer overflow and the message compression. While the IP packets per second are almost the same in experiment 3. In order to see the same behavior of scenario 1, but without packages loss, the same test is repeated in the second scenario. Figure 5.4b shows almost the same IP packets rate between sources and targets, but with a small difference among those experiments. We also complement the discussion by



(a) IP packets without middleware

(b) IP packets with TSI

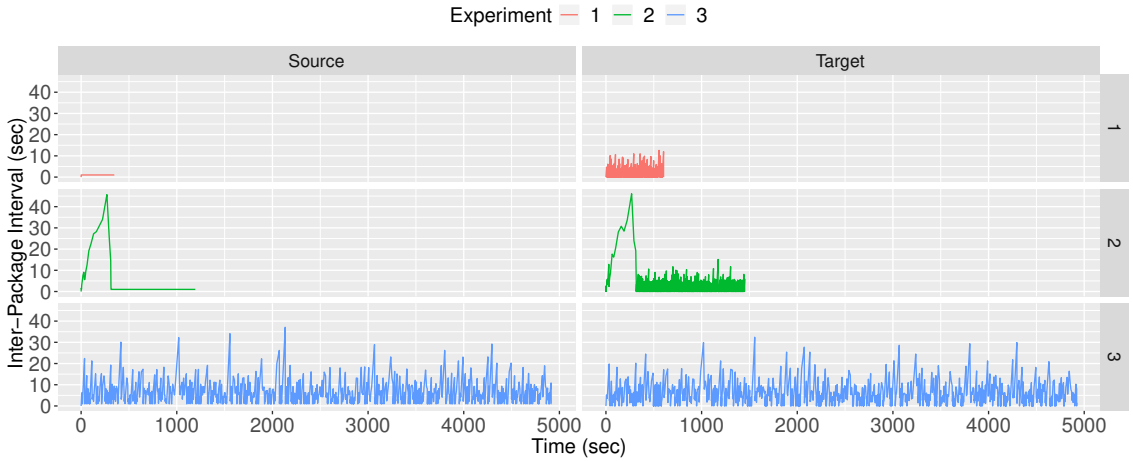
Figure 5.5: IP packets sent and received over time

adding that these values depend on the capacity of the network to send messages to the pipeline.

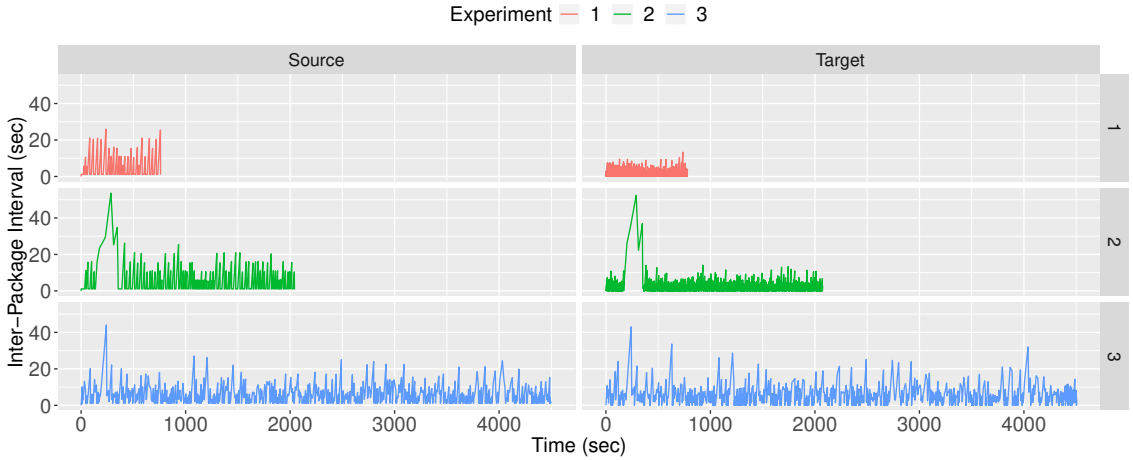
Looking at the number of IP sent and received in a time series, see Figure 5.5, we can observe the difference of line sizes and its inclinations. Comparing the sizes, it is clear to see that we lose packages just in experiments 1 and 2 of scenario 1. On the other hand, the costs to receive all packages sent reflect on the increase of the time (inclination of the curve) among each scenario.

The Figures 5.6a and 5.6b show the value of the IPI over time. Comparing experiment 1 without and with middleware it can be seen how the time between IP packets is fixed and close to zero in the first case (since the source tries to send the information as fast as possible) and how its value varies in the second case. This demonstrates, on the one hand, the constant and problem-free transmission through a Fast Ethernet interface (100 Mb/s) without any kind of intervention to avoid overflow and on the other hand the attempt on the part of TSI to adapt the packet flow according to network conditions to avoid network congestion. Also, experiment 2 presents an interesting peak in both scenarios. This peak can be related to the burst mode of many messages but needs to be tested in more experiments to analyze their behavior. Therefore it is proposed as a future work with which to improve and understand this fact. Even so, it is possible to observe once again the middleware participation when avoiding bottlenecks by comparing the zones where the Figure 5.6b curve fluctuates against the flat zones of Figure 5.6a. Experiment 3 presents a similar behaviour in both scenarios and its value varies according to the time-window between sequences, the differences between the both are not clear in this case.

To summarize the results analyzed above it can be stated that the use of middleware in tactical network is essential to prevent congestion and loss of information. Comparing both scenarios it can be seen that the use of middleware ensures a robust transmission by sending and receiving the same number of packets. In terms of time spent transmitting and receiving IP packets, the TSI graphs show more time in order to avoid loss of packets. TSI, in particular, shows a good performance for large messages (1000 KB) and rush traffic. The effect of the overflow on the packet transmission and reception rate and the IPI parameter has been verified. As well as the behavior of message compression (GZIP) depending on the size of the messages.



(a) IPI without middleware



(b) IPI with TSI

Figure 5.6: IPI over time at both source and target

5.2.1 Simulations: creating patterns of QoS-constrained dataflows

In this section, we discuss simulation results creating different patters of message priority using the stochastic models implemented in our tool. The set up we used for each simulation is summarized in Table 5.2. The probability matrices used in Markov pattern, first row of Table 5.2, are defined in 5.1. The other patterns were created using different approaches. In the following Figures 5.7 to 5.12 the differences between the various priority patterns are illustrated. We generated an array of 100 messages with priorities from 0 to 3 to visually represent the random result of patterns and the messages ordered after processing. In both figures, the left column is shown the chain of unsorted messages while the right column shows them sorted by priority.

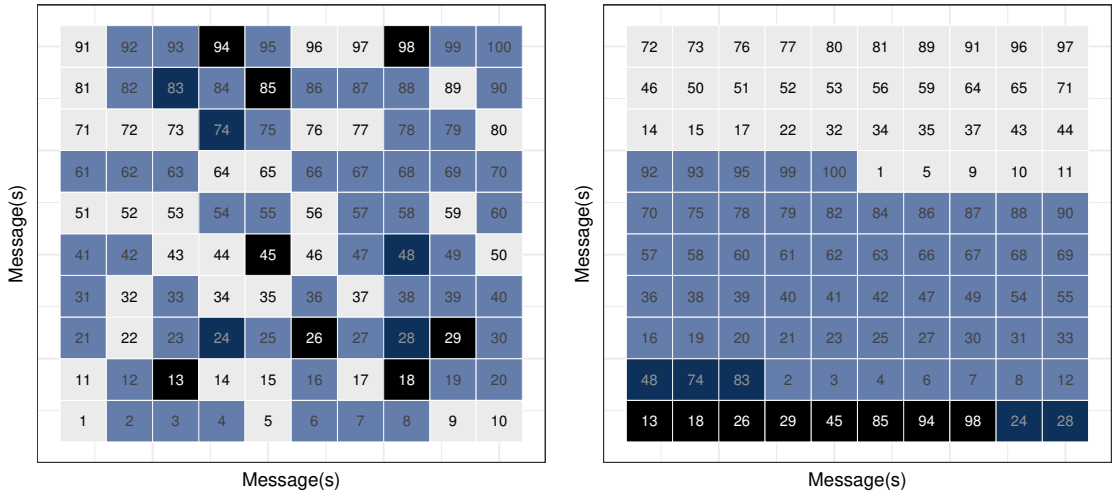
$$A_1 \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} .05 & .05 & .5 & .4 \\ .05 & .05 & .5 & .4 \\ .05 & .05 & .5 & .4 \\ .05 & .05 & .5 & .4 \end{pmatrix} \end{array} \quad A_2 \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} .2 & .2 & .45 & .15 \\ .2 & .2 & .45 & .15 \\ .2 & .2 & .45 & .15 \\ .2 & .2 & .45 & .15 \end{pmatrix} \end{array} \quad A_3 \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} .35 & .4 & .15 & .1 \\ .35 & .4 & .15 & .1 \\ .35 & .4 & .15 & .1 \\ .35 & .4 & .15 & .1 \end{pmatrix} \end{array} \tag{5.1}$$

Pattern	N° of priorities	Priority	Probability	N° of generated messages
Markov	4	Chain (0,1,2,3)	-	100
Msg behaviour	4	Chain (0,1,2,3,0,1,0,2)	-	100
Msg prob.	4	Chain (0,1,2,3)	25 %	100
By default	4	Chain (0,1,2,3)	-	100

Table 5.2: Configuration of the stochastic models used to generate the plots

Figures 5.7 to 5.9 represents those patterns based on Markov (probability matrices 5.1). As is introduced in Chapter 4, the darker the color of the ordered message, the higher priority it will have in the message queue and each matrix seeks to represent from a more stable scenario (A_1) to the most chaotic (A_3).

One method to evaluate the degree of stability of a scenario is to count the number of flash (black squares) priority messages present in the sequence, as they are the most likely to cause problems in the message queue. In Figures 5.7a and 5.7b it can be observed that using matrices with significant differences between the probabilities of change of state, the scenarios are calm since the number of messages with high priority is not high. The opposite effect is seen in Figures 5.9a and 5.9b where the probabilities of the generator matrix used are higher and the message chain shows up to 37 flash priority messages. The intermediate point of view is shown in Figures 5.8a, 5.8b where the number of high priority messages is 23. Notice that these sequences can vary up to $\binom{100}{4} = 3.921.225$ combinations, allowing our tool to generate an ever-change message pattern.

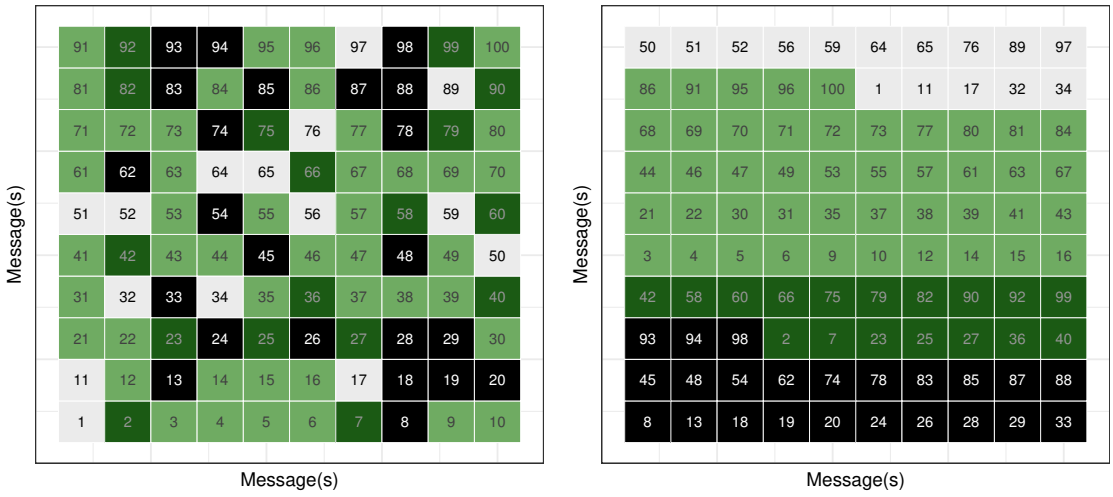


(a) Unsorted: blue sequence

(b) Sorted: blue sequence

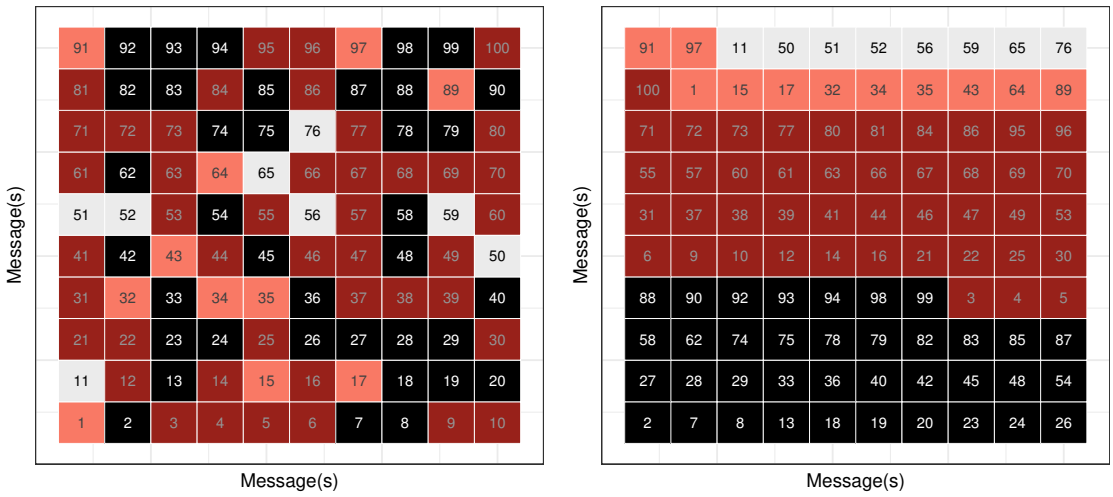
Figure 5.7: Sequence of messages following the priority patterns A_1

The behavior of the rest of the priority patterns offered by our tool also depends on the configuration and scenario desired by the user, but they use different models. The pattern based on the priorities offered by the user, Figures 5.10a and 5.10b, show in our simulations a behavior similar to that of Figures 5.7 to 5.9, since it also relates to Markov in the generation of messages. However, this one uses the message chain provided by the user to build the probability matrix and then generate the sequence while the previous one directly makes use of the probability matrix. In this way it is possible to analyse, for example, the effect of relaunching operations



(a) Unsorted: green sequence (b) Sorted: green sequence

Figure 5.8: Sequence of messages following the priority patterns A_2



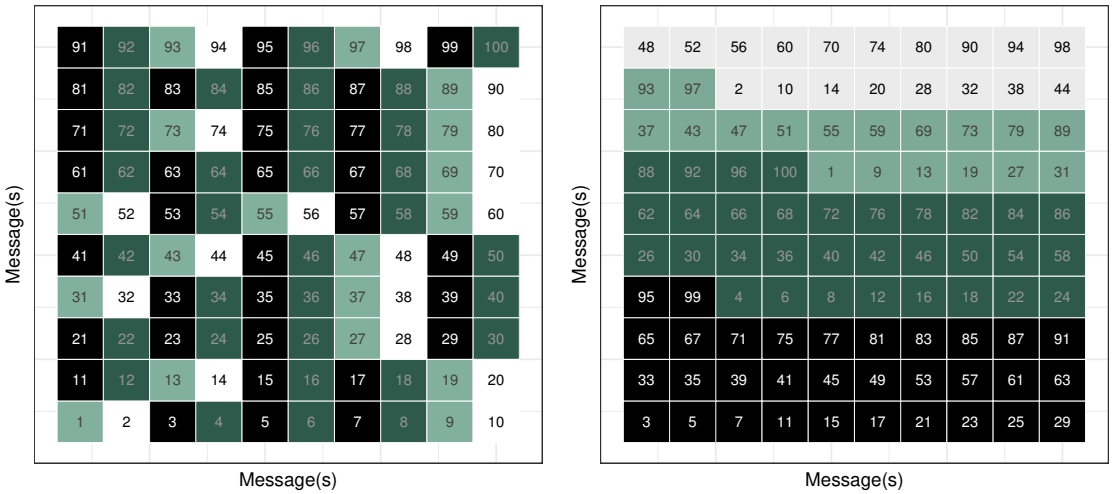
(a) Unsorted: red sequence (b) Sorted: red sequence

Figure 5.9: Sequence of messages following the priority patterns A_3

already experienced in a new network.

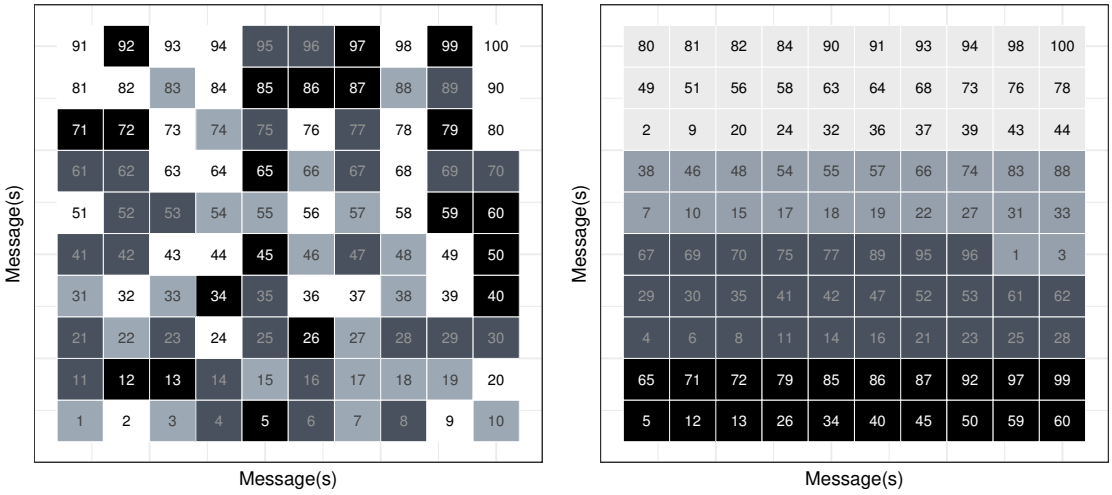
The probability pattern, showed in Figures 5.11a and 5.11b, may behave one way or another depending on the configured values, if a high probability is set to flash priority messages the scenario will be more challenging and vice-versa. Based on observations of real scenarios the user can manage these probabilities to suit his needs. For instance, the user can simulate an evacuation situation after a natural disaster by setting a high probability for messages from the medical service (e.g. 60 %) and readjusting the rest of the services to lower priorities (e.g. OAS 10 %, video 10 %, picture 7 % and FFT 3 %).

Finally, Figures 5.12a and 5.12b exhibit the default pattern that allows creating a situation modifiable by the user simply by varying the value of the parameters in the GUI. The user is free to define the sequence of messages he wants based on artificial situations as well as previous experiences. The Figure 5.12 represents a situation in which a message of each priority is generated sequentially but the user



(a) Unsorted: priority behavior chain (b) Sorted: priority behavior chain

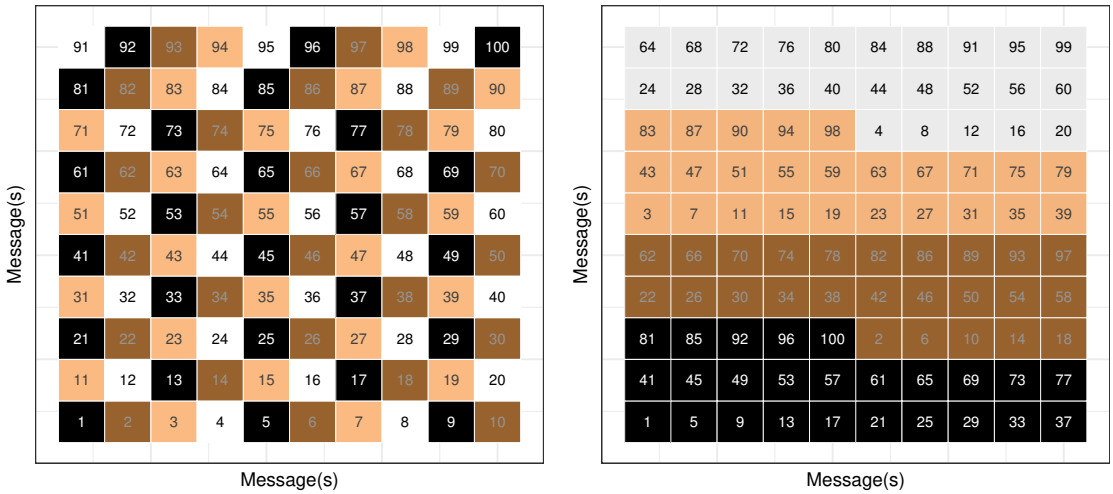
Figure 5.10: Sequence of messages following the priority patterns based on Markov Chain



(a) Unsorted: priority message prob. chain (b) Sorted: priority message prob. chain

Figure 5.11: Sequence of messages following the priority patterns defined by message probability

could emulate, for example, one in which three out of four messages are of maximum priority to try to challenge the performance bounds of a tactical sytem.



(a) Unsorted: priority by default chain

(b) Sorted: priority by default chain

Figure 5.12: Sequence of messages following the priority patterns defined by default

5.3 Final remarks

The experiments carried out aimed to evaluate the functionalities of our tool that generates QoS-constrained dataflows and seeks to test the store-and-forward mechanisms of the tactical system. The performance bounds of the tactical system have been stressed with three different types of traffic. The first consists of a large message (1000 KB) and the next two of 1000 small messages (1 KB each). In turn, these last two experiments sent in burst mode and with a time-window based on an exponential distribution, respectively. Finally, simulations with different priority patterns are shown in order to demonstrate that it is possible to use them in future experiments. With these experiments, we have demonstrated the negative influence of overflow on the radio buffer and the IP packet rate. And how the middleware acts in this case to deal with congestion, among other ways, introducing a time between IP packets to give the system enough time to process the information. We can also say that we achieved our goal by designing this tool.

Chapter 6

Conclusion

The design of real scenarios in the field of tactical networks has been of interest over the years as it allows to analyze, improve and test the tactical systems that will be used in the battlefields. This challenge can be divided into two distinct problems, the changing behavior of users and the constantly changing conditions of the network. Focusing on the user application layer, the current literature shows a lack of randomness in the traffic flows used in the experiments. In this thesis we have implemented a tool called "message benchmark" that generates QoS-constrained data-flows to simulate the exchange of information between different command and control services using the publish/subscribe messaging pattern. For this purpose, a GUI was developed to facilitate the modification of parameters related to the messages, their priorities, and time-windows and simulate different scenarios. The implementation is based on the use of mathematical distributions to introduce the element of randomness in terms of priority and time-window of the messages that are sent with the publish/subscribe pattern. Thus, it is possible to simulate, for example, the exchange of VoIP data by configuring the tool to send a chain of a big number of messages in a short period of time. In this way, by analyzing the effect of the traffic injected into the network, the store-and-forward- mechanisms can be evaluated.

The experiments have been carried out in two scenarios, the first of them to test the operation of the tool without the middleware TSI and the second of them with the middleware. Both scenarios consisted of three tests in which a message of 1000 KB was sent without time-window, 1000 messages of 1 Kb without time-window and 1000 messages of 1KB with time-windows based on an exponential distribution of mean five. With these experiments, we have tested how overflow and the type of traffic transmitted (depending on size and delivery time) affect the radio buffer and IP packet rate. Secondly, we also studied how the middleware acts in these cases to deal with congestion concerning the radio buffer and the IPI introduced.

As a future development of the present thesis, the evaluation of the store-and-forward mechanisms can be carried out using the message benchmark tool that simulates the user layer with the work done in [23] that simulates ever-changing network conditions. In order to obtain results of the problem $A|B$ mentioned earlier in the introductory chapters. Besides, as proposed in Chapter 5, more experiments can be performed to understand the peak shown by the IPI in Figure 5.6 in the second experiment. Finally, another line of work can be to scale the testbed using networks with a greater number of nodes. In this case, a potential improvement

would be to centralize our tool to control and synchronize the traffic flow of each node from a single management node.

Appendix A

Ethical, economic, social and environmental aspects

A.1 Introduction

The project designed for this thesis is located within the sectors of telecommunications and telematics engineering as it is directly related to tactical networks. This work, among other objectives, seeks to promote the experiments and development of middlewares and other research areas providing an environment close to reality in tactical networks. The implementation of a service to create a sequence of QoS-constrained messages and test store-and-forward mechanisms in tactical networks contributes to the scientific community and society. Other groups of interest related to the project are developers, private and public companies. Both companies looking to test their middleware or networks and governments interested in improving their tactical systems can benefit from this service.

As it is indicated in the Introduction chapter, the organizational and strategic scope in which this work is placed belongs to the European TACTICS project. Specifically in the expansion of the work done by Lopes et al. [11] in the area of analysis and testing of tactical middlewares based in web services. The main institution involved is the Fraunhofer Institute for Communication, Information Processing and Ergonomics (FKIE).

In relation to the life cycle, the first phase consisted of a preliminary study of related works and the definition of the problem to be solved. Afterwards, the requirements and scope are described to launch a hypothesis and give way to the design. In this phase, the tools to be used are decided and the implementation of the prototype begins. After this, tests and experiments are performed to analyse errors and improvements and check whether the hypothesis is erroneous or not. Finally, the cycle is repeated in the case of new proposals or decisions that can be improved and, in the case of success, the maintenance phase is carried out.

A.2 Description of relevant impacts related to the project

In this section the most relevant repercussions of the project are analysed in the ethical, social, economic and environmental context. The evaluation seeks to describe

the main impacts (positive and negative), problems or aspects previously identified as related to the project. The relevant influences are categorised considering the area to which they belong, therefore we can distinguish:

- **Technology**: the project will improve future experiments related to the tactical networks. This work allows other researchers to use a real scenario, improving the quality of their final proposals and as a consequence the development of better military networks.

- **Social**: in terms of social issues and as a direct result of the one mentioned in technology, this will make it possible to work in better conditions to the forces involved in natural disasters or battlefields. In addition to military environments, this proposal can be used in other networks related to the exchange of messages with different priority. The sectors benefiting from this are the forces involved in military operations, people requiring their help (e.g. in a natural disaster) and scientists and developers of similar solutions or proposals. As a negative impact, there may be a group positioned against the improvement of tactical networks for war or territorial purposes.

- **Economic**: the economic impact of the work implies significant savings in costs since it is not necessary to carry out a deployment of forces to perform the experiments in the user layer. The approximate budget for the deployment of an individual soldier was approximately 40.441 €in 2002 (considering training, personnel costs and salary) [60]. Furthermore, the set up of the testbed can be done in two different ways as experienced in this Master Thesis: (1) with military radios, switches and real nodes or (2) in a virtualized network. Both solutions suppose a lower cost than the use of forces, requiring an approximate investment of 143.222 €(see Appendix B) in the first case and some computers (e.g. three) without the need of high technical specifications in the second case.

- **Environmental**: the main environmental impact is the energy consumption related to the radios, nodes and computers used in the laboratory. The batteries of the radios have to be replaced and recharged every 24 hours approximately and they are made of lithium-ion. The manufacture of batteries with chemical products together with their high consumption, can pollute and harm the environment. In addition they contain a flammable electrolyte under pressure.

- **Legal**: any project involved in tactical networks must ensure security in the processing of information. The consequences of loss of information or intrusion can cause serious consequences. In this work some messages are prioritized over others in order to speed up their processing and response. But security depends mainly on the sender and its use, in this case, of SOAP-SEC with web services.

- **Operational**: in the service there has not been designed any system that protect the messages as the transport protocol used in tactical networks is normally UDP and it does not ensure the arrival of them.

As a conclusion after analyzing the impacts of the project, the most significant are those related to the areas of economy and environment. These are discussed in the following subsection.

A.3 Detailed analysis of the main impacts

The details related to the economic budget are broken down in Appendix B. As can be observed, most of the costs related to materials are due to the use of military radios, representing 94,5 % of the total. Comparing the total price of the project calculated in Appendix B of 143.222 €with the estimated price per soldier according to CNBC, this budget would only cover the contract of three soldiers to which would have to be added the cost of military vehicles and radios. This shows the savings involved in carrying out this type of work that seeks to create real scenarios compared to carrying out experiments in real fields.

The environmental impact of the project lies mainly in the use of lithium-ion batteries in military radios. The huge demand for this type of batteries (used in electronic devices and the automotive industry) at a global level causes the exhaust of natural resources because of intense drilling operations to extract the lithium. Also, the chemical used to extract the lithium from the ground is capable of infiltrating nearby rivers, streams and water supplies. These batteries need to be recharged every 24 hours approximately with the electrical consumption that this supposes. They can overheat to the point of exploding because they are made of flammable materials that make them prone to detonations or fires. This makes it essential to provide electronic circuits that control their temperature at all times.

A.4 Conclusions

Taking into account the aspects mentioned above it can be observed that the most important impact of the project is the economic one, but at the same time the environmental one must be considered. The cost savings are significant but also the consumption of the radios added to nodes and computers supposes a high energetic consumption.

Appendix B

Economical budget

This annex summarizes the costs associated with the completion of this thesis. It then covers aspects associated with materials, professional fees and total costs.

B.1 Cost of materials

The cost associated with the materials is shown in Table B.1. This table lists the hardware used during this investigation such as laptos, radios and network switches. The software packages used includes RStudio, Eclipse IDE, Java SDK and Linux as operating system, which are open source.

Material	**Unit cost (€)**	**Total cost (€)**
Laptop + Software	1.500 €	3.000 €
PR4G Radio	60.000 €	120.000 €
Fanless Barebone Router	144 €	288 €
Uniquiti Tough Switch	85 €	170 €
TOTAL		123.458 €

Table B.1: Costs of materials

B.2 Professional fees

This section considers expenses related to professional fees involved in the development of the project. The total cost of the fees is shown below in Table B.2.

Description	**Hours spent (h)**	**Salary (€/h)**	**Total cost (€)**
Engineer	810 h	20 €/h	16.200 €
TOTAL			16.200 €

Table B.2: Professional fees

B.3 Total costs

The total expenses of the project are compiled in Table B.3 and are the sum of the two previous ones. It includes the salary taxes standardized in Germany.

Description	**Costs (€)**
Cost of materials	123.458 €
Professional fees	19.764 €
TOTAL	143.222 €

Table B.3: Total costs

Appendix C

Scripts

Data: Desired number of messages, time-window sequence and topic characteristics.
Result: This script receives the topic characteristics and user preferences and launches the service.
initialization;
read the number of messages, time-window and topic characteristics;
Function `sendSelectedRows(`*selectedRow*`):`
    This function gets the number of desired intervals and the time window and generates a LinkedHashMap to store the messages. Then it sends the sequence by creating a unique identifier for the publisher (using the host name), sending notifications for the topic, creating a NotificationBroker proxy and sleeping *timeWindows* ms.;
End Function

Algorithm 1: Tactics Script

Data: Probability matrix, service matrix and initial service.
Result: This script generates a dataset of random messages sorted by priority maintaining their original message number.
initialization;
if *data* = *data_sorted* **then**
    `// In case of looking for sorted data`
    startSequence(data, ProbMatrix, ServiceMatrix);
    sortMessages(data);
else
    `// In case of looking for unsorted data`
    startSequence(data, ProbMatrix, ServiceMatrix);
end
Function `startSequence(`*data, A, services*`):`
    This function generates a message dataset using *nextService()* to fill the priorities according to the sequence of services and their probabilities.;
    return *data*;
End Function
Function `nextService(`*dice, probVector*`):`
    This function throws a dice to select the nextService. If the dice value is between the probabilities of two consecutive services then the next service will be the one with the higher value. In other case the next service will be the one with the closest probability.;
    return *serviceIndex*;
End Function
Function `sortMessages(`*data*`):`
    This function sorts the data by priority in ascending order.;
    return *dataSorted*;
End Function

Algorithm 2: Sequence of Messages Script

```
Data: User setting preferences.
Result: This script shows a GUI to allow user interaction and send the
        parameters to Tactics script.
initialization;
if Ecm_button is clicked then
    read the user election;
    switch module selected do
        case size do
            if Seq or Per message is selected then
                msgSizeGenerator()
            else
                // If the user clicks By default option
                read GUI table
            end
        case priority do
            if Low or Medium or High is selected then
                MarkovMsgGeneratorProbMatrix()
            else if Msg Behavior is selected then
                MarkovMsgGeneratorInputPattern(int priority [])
            else if Msg Prob. is selected then
                ProbSequenceOfMsgGenerator(int desiredMsg, int prob [])
            else // If the user clicks By default option
            ;
            read GUI table
        case time-window do
            if Disitribution is selected then
                show distributions;
                RandonNumberGenerator()
            else
                // If the user clicks static option
                Fix timeWindow;
            end
    end
else
    wait until interaction
end
```

Algorithm 3: Obstacle Alert Script

Acronyms

A Adjustable parameter. 14

ACM Agile Communications Middleware. 13

API Application Programming Interface. 9, 13

ARL ARL Traffic Generation Tool. 14

C2 Command and Control. 2, 5, 7, 24

C3 Consultation, Command and Control. 17

C4ISR Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance. 2

CIS Communication and Information Systems. 8

COI Community of Interest. 8, 25

COP Common Operational Pictures. 25

CPU Central Processing Unit. 25

CWIX Coalition Warrior Interoperability eXploration, eXperimentation, eXamination, eXercise). 3

DDAM Dynamic Detect and Adapt Mechanism. 13

DDR Double Data Rate. 25

FCS Future Core Services. 3

FFT Friendly Force Tracking. 5

FIFO First in, First out. 10

FKIE Fraunhofer FKIE. 24

FTP File Transfer Protocol. 12

GB Gigabyte. 25

GUI Graphical User Interface. 6, 12–14, 24, 26–29, 33, 34, 44, 47

HDD Hard Disk Drive. 25

HF High Frequency. 14

HQ Headquarters. 16

HTTP Hypertext Transfer Protocol. 4

IBM International Business Machines Corporation. 12

ID Identification. 12

IP Internet Protocol. 4, 6, 14, 37, 39–41, 46, 47

IPI Internet-Packet-Interval. 6, 18, 37, 41, 42, 47

IT Information Technology. 11

JEMS JBoss Enterprise Middleware Stack. 11

JMS Java Message Service. 1, 9–12, 14, 15

JNDI Java Naming and Directory Interface. 12

LAN Local Area Network. 14

MANE Mobile Ad-hoc Network Emulator. 13

MANET Mobile Adhoc Network. 13, 14

MB Megabyte. 25

MGEN Multi-Generator. 14

MILCOM Military Communications. 13

MOM Message-Oriented Middleware. 9–13

MQ Message Queue. 12

NAS Network Awareness Service. 13

NATO North Atlantic Treaty Organization. 3, 4

NEC Network Enabled Capabilities. 4

NNEC FS Network Enabled Capability Feasibility Study. 4, 15

NRL Naval Research Laboratory. 14

OAS Obstacle Alert Service. 24, 25

OLSR Optimized Link State Routing. 14

P Predefined parameter. 14

PoE Power over Ethernet. 25

PROTEAN PROTocol Engineering Advanced Networking. 14

QAM QoS-aware Adaptive Middleware. 13

QoS Quality of Service. 2–6, 11, 22, 25, 36, 47

RAM Random Access Memory. 25

REST Representational State Transfer. 12

RF Radio Frequency. 25

RPM Revolutions Per Minute. 25

SatCom Satellite Communications. 2, 14, 16, 25

SDRAM Synchronous Dynamic Random-Access Memory. 25

SOA Service Oriented Architecture. 4, 10, 11, 15, 17

SOAP Simple Object Access Protocol. 4, 11, 12

SOAP-SEC SOAP Security. 11

TACTICS Tactical Service-Oriented Architecture. 3, 6, 16, 17, 22, 24, 25, 36

TAKE Tactical Ad-hoc networK Emulation. 14

TB Terabyte. 25

TCP Transmission Control Protocol. 12, 14

TEN Tactical Edge Networks. 14

TIB/RV Tibco Rendezvous. 13

ToE Time of Expire. 5

TSI Tactical Service-Oriented Infrastructure. 3, 16, 17, 24, 36–42, 47

UDDI Universal Description Discovery and Integration. 11

UDP User Datagram Protocol. 3, 14, 37, 39, 50

UHF Ultra High Frequency. 2, 16

UML Unified Modeling Language. 6, 29, 35

V Volt. 25

VHF Very High Frequency. 2, 6, 16, 23, 25, 34

VLAN Virtual LAN. 16, 25

VoIP Voice over IP. 14, 37, 47

W Watt. 25

WS Web Services. 10, 11

WSDL Web Services Description Language. 11

XML Extensible Markup Language. 4, 11

Bibliography

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

[2] Rotermund, M., "Consultation, command and control board (c3b) c3 taxonomy baseline 2.0." `https://www.nato.int/nato_static_fl2014/assets/pdf/pdf_2018_08/20180801_180801-ac322-d_2016_0017-c3t.pdf`. [Online; accessed 2019-02-20].

[3] IBM Developer, "Choosing among jca, jms, and web services for eai." `https://www.ibm.com/developerworks/webservices/library/ws-jcajms/index.html`. [Online; accessed 2019-07-10].

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

[5] G. Elmasry, "A comparative review of commercial vs. tactical wireless networks," *IEEE Communications Magazine*, vol. 48, pp. 54–59, October 2010.

[6] NATO, "What is nato?." `https://www.nato.int/nato-welcome/index.html`. [Online; accessed 2019-04-25].

[7] NATO, "Coalition warrior interoperability exercise." `https://www.act.nato.int/cwix`. [Online; accessed 2019-07-07].

[8] A. Diefenbach, T. Ginzler, S. McLaughlin, J. Sliwa, T. A. Lampe, and C. Prasse, "TACTICS TSI architecture: A european reference architecture for tactical SOA," in *International Conference on Military Communications and Information Systems (ICMCIS)*, pp. 1–8, May 2016.

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

[10] R. R. F. Lopes, P. H. Balaraju, and P. Sevenich, "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.

[11] R. R. F. Lopes, P. H. Balaraju, and P. Sevenich, "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.

[12] R. R. F. Lopes, A. Viidanoja, M. Lhotellier, M. Mazurkiewicz, G. Melis, A. Diefenbach, T. Ginzler, and N. Jansen, "Trade-off analysis of a service-oriented and hierarchical queuing mechanism," in *IEEE 16th International Symposium on Network Computing and Applications (NCA)*, pp. 1–4, Oct 2017.

[13] N. Jansen, D. Krämer, C. Barz, J. Niewiejska, and M. Spielmann, "Middleware for coordinating a tactical router with SOA services," in *International Conference on Military Communications and Information Systems (ICMCIS)*, pp. 1–7, May 2015.

[14] M. Manso, J. M. A. Calero, C. Barz, T. H. Bloebaum, K. Chan, N. Jansen, F. T. Johnsen, G. Markarian, P. P. Meiler, I. Owens, J. Sliwa, and Q. Wang, "SOA and wireless mobile networks in the tactical domain: Results from experiments," in *IEEE Military Communications Conference (MILCOM)*, pp. 593–598, Oct 2015.

[15] R. R. F. Lopes, M. Nieminen, A. Viidanoja, and S. D. Wolthusen, "Reactive/proactive connectivity management in a tactical service-oriented infrastructure," in *International Conference on Military Communications and Information Systems (ICMCIS)*, pp. 1–8, May 2017.

[16] J. Nightingale, Q. Wang, J. M. A. Calero, I. Owens, F. T. Johnsen, T. H. Bloebaum, and M. Manso, "Reliable full motion video services in disadvantaged tactical radio networks," in *International Conference on Military Communications and Information Systems (ICMCIS)*, pp. 1–9, May 2016.

[17] K. Wrona, S. Oudkerk, A. Armando, S. Ranise, R. Traverso, L. Ferrari, and R. McEvoy, "Assisted content-based labelling and classification of documents," in *International Conference on Military Communications and Information Systems (ICMCIS)*, pp. 1–7, May 2016.

[18] J. Sliwa and B. Jasiul, "Efficiency of dynamic content adaptation based on semantic description of web service call context," in *MILCOM 2012 - 2012 IEEE Military Communications Conference*, pp. 1–6, Oct 2012.

[19] K. Lund, E. Skjervold, F. Johnsen, T. Hafsøe, and A. Eggen, "Robust web services in heterogeneous military networks," *IEEE Communications Magazine*, vol. 48, pp. 78–83, October 2010.

[20] Buckman. T., "NATO network enabled capability feasibility study executive summary : Version 2.0." `http://www.dodccrp.org/files/nnec_fs_executive_summary_2.0_nu.pdf`. [Online; accessed 2019-03-11].

[21] K. Lund, A. Eggen, D. Hadzic, T. Hafsoe, and F. T. Johnsen, "Using web services to realize service oriented architecture in military communication networks," *IEEE Communications Magazine*, vol. 45, pp. 47–53, October 2007.

[22] G. L. Eric Newcomer, *Understanding SOA with Web Services*. Independent technology guides, Addison-Wesley, 2005.

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

[24] J. R. A. Marius S. Vassiliou, David S. Alberts, *C2 Re-envisioned: The Future of the Enterprise*. CRC Press, 1 ed., 2014.

[25] A. Ghosh, S. Li, C. J. Chiang, R. Chadha, K. Moeltner, S. Ali, Y. Kumar, and R. Bauer, "Qos-aware adaptive middleware (qam) for tactical manet applications," in *MILITARY COMMUNICATIONS CONFERENCE (MILCOM)*, pp. 178–183, Oct 2010.

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

[27] RedHat, "¿qué es middleware?." `https://www.redhat.com/es/topics/middleware/what-is-middleware`. [Online; accessed 2019-07-10].

[28] D. Oberle, A. Eberhart, S. Staab, and R. Volz, "Developing and managing software components in an ontology-based application server," in *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '04, (Berlin, Heidelberg), pp. 459–477, Springer-Verlag, 2004.

[29] A. Grant, M. Antonioletti, A. C. Hume, A. Krause, B. Dobrzelecki, M. J. Jackson, M. Parsons, M. P. Atkinson, and E. Theocharopoulos, "Ogsa-dai: Middleware for data integration: Selected applications," in *2008 IEEE Fourth International Conference on eScience*, pp. 343–343, Dec 2008.

[30] B. Dobrzelecki, A. Krause, A. C. Hume, A. Grant, M. Antonioletti, T. Y. Alemu, M. Atkinson, M. Jackson, and E. Theocharopoulos, "Integrating distributed data sources with ogsa-dai dqp and views," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1926, pp. 4133–4145, 2010.

[31] S. Vinoski, "Where is middleware," *IEEE Internet Computing*, vol. 6, pp. 83–85, March 2002.

[32] P. A. Bernstein, "Middleware: A model for distributed system services," *Commun. ACM*, vol. 39, pp. 86–98, 1996.

[33] M. X. Y. T. M. Q. S. L. W. C. Yoshikawa, M., *Database Systems for Advanced Applications*. Elsevier, 2010.

[34] JBoss, "Chapter 7. generating performance benchmark results." `https://docs.jboss.org/jbossmessaging/docs/guide-1.0.1.SP5/html/performance.html`. [Online; accessed 2019-07-19].

[35] Apache, "Jmeter performance test." https://activemq.apache.org/jmeter-performance-tests.html. [Online; accessed 2019-07-19].

[36] IBM, "Ibm mq c performance harness." https://github.com/ibm-messaging/mq-cph/blob/master/cph.pdf. [Online; accessed 2019-07-19].

[37] JBoss, "Chapter 2. introduction." https://docs.jboss.org/jbossmessaging/docs/guide-1.0.1.SP5/html/introduction.html. [Online; accessed 2019-07-19].

[38] Apache, "Apache jmeter." http://jmeter.apache.org/. [Online; accessed 2019-07-19].

[39] IBM Community, "Mqdev blog." https://www.ibm.com/developerworks/community/blogs/messaging/entry/MQ_C_Performance_Harness_Released_on_GitHub?lang=en. [Online; accessed 2019-07-19].

[40] K. Sachs, S. Appel, S. Kounev, and A. Buchmann, "Benchmarking publish/subscribe-based messaging systems," vol. 6193, pp. 203–214, 04 2010.

[41] The linux foundation, "The openmessaging benchmark framework." http://openmessaging.cloud/docs/benchmarks/. [Online; accessed 2019-07-20].

[42] Ta Chen, S. Eswaran, M. A. Kaplan, S. Samtani, D. Shur, J. Sucec, and L. Wong, "Enhancing application performance with network awareness in tactical networks," in *2011 - MILCOM 2011 Military Communications Conference*, pp. 1158–1163, Nov 2011.

[43] P. Maheshwari and M. Pang, "Benchmarking message-oriented middleware: Tib/rv versus sonicmq," *Concurrency - Practice and Experience*, vol. 17, pp. 1507–1526, 10 2005.

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

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

[46] US Army Research Laboratory., "Traffic generation tool.." https://www.arl.army.mil/www/default.cfm?page=2490. [Online; accessed 2019-07-26].

[47] US Naval Research Laboratory., "Multi-generator (mgen).." https://www.nrl.navy.mil/itd/ncs/products/mgen. [Online; accessed 2019-07-26].

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

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

[50] Thales Group, "PR4G f@stnet product family." https://www.thalesgroup.com/en/worldwide/defence/pr4g-fstnet-product-family. [Online; accessed 2019-04-08].

[51] Dell, "Inspiron 15 3000 laptop." https://www.dell.com/ve/p/inspiron-15-3558-laptop/pd. [Online; accessed 2019-04-08].

[52] Amazon, "Bar bones firewall pc mi19 n." https://www.amazon.de/Barebones-Firewall-Mi19N-Intel-Celeron/dp/B01MSW5RXS. [Online; accessed 2019-04-08].

[53] Baltic Networks Inc., "Tough switch poe: Datasheet." https://www.balticnetworks.com/docs/TOUGHSwitch_PoE_DS.pdf. [Online; accessed 2019-04-08].

[54] D. Bertsekas and J. Tsitsiklis, *Introduction to Probability*. Athena Scientific books, Athena Scientific, 2002.

[55] S. M. Ross, *Introduction to Probability Models - 10th ed.* Elsevier, 2010.

[56] The R Foundation., "The r project for statistical computing.." https://www.r-project.org/. [Online; accessed 2019-08-06].

[57] Red Hat Ansible., "Ansible is simple it automation.." https://www.ansible.com/. [Online; accessed 2019-08-06].

[58] US Naval Research Laboratory., "Mgen user's and reference guide version 5.0." https://downloads.pf.itd.nrl.navy.mil/docs/mgen/mgen.html. [Online; accessed 2019-08-16].

[59] GZIP, "Gzip home." https://www.gzip.org/. [Online; accessed 2019-08-14].

[60] Diana Olick - CNBC, "An army of one carries a high price." http://www.nbcnews.com/id/3072945/t/army-one-carries-high-price/.XT8Xv-jHxPY. [Online; accessed 2019-08-16].