Multicast Communication for Increased Data Exchange in Data-Intensive Distributed Systems

FELICIA SCHNELL
Abstract

Modern applications are required to handle and communicate an increasing amount of data. Meanwhile, distributed systems containing multiple computationally weak components becomes more common, resulting in a problematic situation. Choosing among communication strategies, used for delivering message between entities, therefore becomes crucial in order to efficiently utilize available resources.

Systems where identical data is delivered to many recipients are common nowadays, but may apply an underlying communication strategy based on direct interaction between sender and receiver which is insufficient. Multicasting refers to a technique for group communication where messages can be distributed to participating nodes in a single transmission. This technique is developed to circumvent the problem of high workload on sender side and redundant traffic in the network, and constitutes the focus for this thesis.

Within the area of Electronic Warfare and self-protection systems, time constitutes a critical aspect in order to provide relevant information for decision making. Self-protection systems developed by Saab, used in military aircrafts, must provide situational awareness to guarantee that correct decisions can be made at the right time. With more advanced systems, where the amount of data needed to be transmitted increases, the need of fast communication is essential to achieve quality of service. This thesis investigates how the deployment of multicast, in a distributed data-intensive system, could prepare a system for increased data exchange.

The result is a communication design which allows for the system to distribute messages to a group of receivers with less effort from the sender and with reduced redundant traffic transferred over the same link. Comparative measurements are conducted between the new implementation and the old system. The result of the evaluation shows that the multicast solution both can decrease the time for message handling as well as the workload on endpoints significantly.

Keywords: Distributed systems, Message handling, Multicast, Publish/Subscribe, Electronic Warfare
Abstract

Nutidens applikationer måste kunna hantera och kommunicera en ökad datamängd. Samtidigt har distribuerade system bestående av många beräkningsmässigt svaga enheter blivit allt mer vanligt, vilket är problematiskt. Valet av kommunikationsstrategi, för att leverera data mellan enheter i ett system, är därför av stor betydelse för att uppnå effektivt utnyttjande av tillgängliga resurser.

System där identisk information ska distribueras till flertalet mottagare är vanligt förekommande idag. Den underliggande kommunikationsstrategin som används kan dock baseras på direkt interaktion mellan sändare och mottagare vilket är ineffektivt. Multicast (Flersändning) syftar till ett samlingsbegrepp inom datorkommunikation baserat på gruppsändning av information. Denna teknik är utvecklad för att kringgå problematiken med hög belastning på sändarsidan och dessutom minska belastningen på nätverket, och utgör fokus för detta arbete.

Inom telekrigföring och självskyddssystem utgör tiden en betydande faktor för att kunna tillhandahålla relevant information som kan stödja beslutsfattning. För självskyddssystem utvecklade av Saab, vilka används i militärflygplan, är situationsmedvetenhet av stor betydelse då det möjliggör för att korrekta beslut kan tas vid rätt tidpunkt. Genom utvecklingen av mer avancerade system, där mängden meddelanden som måste passera genom nätverket ökar, tillkommer höga krav på snabb kommunikation för att kunna åstadkomma kvalité. Denna uppsatsrapport undersöker hur införandet av multicast, i ett dataintensivt distribuerat system, kan förbereda ett system för ökat datautbyte.

Arbetet har resulterat i en kommunikationsdesign som gör det möjligt för systemet att distribuera meddelanden till grupp av mottagare med minskad belastning på sändarsidan och mindre redundant trafik på de utgående länkarna. Jämförandet mätningar har gjorts mellan den nya implementationen och det gamla systemet. Resultaten visar att multicast-lösningen både kan reducera tiden för meddelande hantering samt belastningen på ändnoder avsevärt.

Nyckelord: Distribuerade system, Meddelandehantering, Multicast, Publish/Subscribe, Telekrigföring
Acknowledgement

This thesis is the result of a degree project conducted as a final part of the Master’s Programme in Software Engineering of Distributed Systems at KTH Royal Institute of Technology during spring 2018. The project has been carried out on the behalf of Saab Surveillance in Järfälla, Stockholm.

I would like to express my sincere gratitude to my company supervisor Fredrik for his continuous support and commitment during the project. Also, I wish to thank my examiner Anne Håkansson for guiding me through the project process. Furthermore, I would like to address a special thank you to Mikael Mölder for giving me valuable feedback concerning my thesis, and for all the fun we have had in the last five years.

Finally, I would like to thank the other students conducting their master thesis’ at Saab; Laura, Sofia J, Fanny, Araxi, Sofia B, Jorge, Eric and Johan. Sharing this experience with you has been a pleasure and I really appreciated it.
# Table of Contents

1 Introduction ........................................................................................................... 1  
  1.1 Background .......................................................................................................... 1  
  1.2 Problem ................................................................................................................ 2  
  1.3 Purpose .................................................................................................................. 3  
  1.4 Goal ....................................................................................................................... 3  
    1.4.1 Benefits, Ethics and Sustainability ................................................................... 4  
  1.5 Research methodology ......................................................................................... 4  
  1.6 Stakeholder ......................................................................................................... 6  
  1.7 Delimitations ....................................................................................................... 7  
  1.8 Outline ................................................................................................................ 7  
2 Point-to-multipoint communication in distributed systems ......................... 8  
  2.1 Concept of data communication ....................................................................... 8  
    2.1.1 Transport protocols ...................................................................................... 9  
    2.1.2 Network and Data Link layer ...................................................................... 9  
  2.2 Publish/Subscribe communication pattern ..................................................... 10  
  2.3 Multicast, unicast and broadcast ....................................................................... 10  
  2.4 Multicast addressing ......................................................................................... 12  
  2.5 Multicast routing ................................................................................................ 12  
  2.6 Related work ...................................................................................................... 13  
3 Methodology .......................................................................................................... 15  
  3.1 Data collection methods .................................................................................... 16  
    3.1.1 Documents study ....................................................................................... 16  
    3.1.2 Interviews .................................................................................................. 16  
    3.1.3 Experiments .............................................................................................. 17  
  3.2 Data analysis ...................................................................................................... 18  
  3.3 Working process ................................................................................................ 18  
    3.3.1 Pre-study ..................................................................................................... 19  
    3.3.2 Requirements elicitation .......................................................................... 20  
    3.3.3 System development ................................................................................ 20  
    3.3.4 Evaluation ................................................................................................ 21  
4 Pre-study of existing system design .................................................................. 22  
  4.1 System overview ................................................................................................ 22  
  4.2 Communication model ...................................................................................... 23  
  4.3 Message delivery ................................................................................................ 25  
  4.4 System limitations .............................................................................................. 25  
5 Applying multicast communication in a data-intensive distributed system .......... 27  
  5.1 System requirements ........................................................................................ 27  
  5.2 System implementation .................................................................................... 28  
    5.2.1 Overview .................................................................................................... 29  
    5.2.2 Data distribution service .......................................................................... 29  
    5.2.3 Multicast groups and addressing .............................................................. 30  
    5.2.4 Communication sockets .......................................................................... 31  
    5.2.5 Group membership .................................................................................. 32  
    5.2.6 Switch filtering ......................................................................................... 33  
  5.3 Final communication design ............................................................................. 34
1 Introduction

Advancements in computing technology over the years has led to the development of large and data-intensive systems. As systems become larger and more distributed, the amount of data that needs to be communicated within these systems has vastly increased. Additionally, this leads to increased requirements on the inter-process communication (IPC) [1] in order to still be able to provide high quality services.

For many decades, Electronic Warfare (EW) [2] have been an essential part within the area of surveillance where exploitation of the electromagnetic spectrum is used to detect, identify and protect against potential threats. The ability to rapidly gain information about the surroundings and converting it into decisions has become even more important in military defense nowadays [3]. Strategies to influence other actors’ decisions, and thereby protect own decision makers, are constantly evolving which has resulted in advanced and high-technological systems for self-protection. Since systems within this discipline are concerned by time constraints when it comes to collecting, analyzing and acting upon information, the internal communication becomes crucial. Adopting the underlying communication strategy, to handle an increasing exchange of information, is therefore essential in order to meet the requirements of today’s military landscape.

1.1 Background

Along with the technical development, distributed systems have become a fundamental part of the society, providing important infrastructures for information and communication in people’s everyday environment. Several definitions for describing the concept of distributed systems are available [1], [4] but in general it can be said to be a collection of computing entities working together in order to achieve common goals. The communication between entities becomes central in order to coordinate actions which helps in solving the addressed problem and constitutes the heart of any distributed system.

Collaborative networks of embedded devices which collect and share information among each other to perform various tasks exist everywhere nowadays. These networks may be of centralized, decentralized or distributed nature, see Figure 1 [5], describing characteristics for data exchange in the system and have in common that the communication is based on message passing [1]. With new technology constantly influencing the market, applications where the same information is distributed to many recipients, like multimedia streaming [6], have become widely used. Applications applying a publish/subscribe message pattern [7], where data should be selectively
delivered to those with an interest in the current information, is thus common nowadays. How data is transmitted over the network may, however, differ.

![Diagram of network designs]

**Figure 1: Different network designs, adapted from [5]**

Strategies for message delivery over a network is often based on traditional unicast [8], where data is transmitted in a direct manner between sender and receiver. When identical data is supposed to be communicated to several recipients, the technique of unicast may be an insufficient alternative as it requires individual delivery to each entity interested in the certain message.

Multicasting refers to the communication paradigm where delivery of messages goes from one sender to multiple receivers [6]. It has the benefit of providing message delivery to a destination group in a single transmission, as messages are replicated on network elements, i.e. routers and switches. This means that only one message needs to be sent on each link in the network which can decrease the amount of copies on the outgoing link.

In a situation where applications become more data-intensive and systems increase in size, so does the amount of data which need to be communicated in these networks. Furthermore, the situation indicates that the underlying communication approach may have a substantial impact on the overall performance of the system. Examining and wisely choose strategies for information exchange is thus important in order to provide efficiency by better utilization of available resources.

### 1.2 Problem

Increased requirements on computer systems, to manage more complex and demanding computations, has led to the development of large and advanced distributed systems. However, as the gap between data transmission volumes and bandwidth tends to widen [9], [10], efficient methods for data communication are essential.

Considering systems where message delivery goes from one sender to multiple receivers, the problem is that traditional communication based on unicast
causes redundant traffic in a point-to-multipoint scenario, since identical data are transferred over the same link several times. Further, this leads to inefficient use of the network and may cause high load and potential bottlenecks when data transmission increases. As the data-intensity increase in almost every sector, systems based on embedded heterogeneous devices with limited resources will suffer even more from a communication strategy which is not adapted to the purpose of the application. Minimizing congestion in the network and workload on endpoints is thus desirable to be able to increase data exchange and interaction between entities. By reducing the amount of resources used for message handling, the system can be prepared for an increased data flow.

Within industry, where products evolve during long time, systems become increasingly complex. Moreover, the implementation of changes is often a complicated process which may inhibit innovation and the possibilities of adapting products to the latest technology. With the development of advanced and sophisticated systems in the military domain, the amount of data that needs to be communicated within these systems have increased. For self-protection systems used in military aircrafts, which constitutes the case study for this project, the ability to share information among different components in the system becomes crucial. Fast communication with low latency poses significant attributes to support decision making at any given time. The problem is that the communication architecture is not currently adapted to fully manage message distribution of point-to-multipoint nature in an efficient manner. This might lead to high network load, eventually causing latencies affecting quality of the service.

From the problem presented, the main question of the project is formulated as: How can multicast communication be applied in a data-intensive distributed system to support increased data exchange?

1.3 Purpose

The purpose of the thesis is to present the development and evaluation of a point-to-multipoint communication strategy for a data-intensive distributed system. More specifically, the thesis aims to describe the process of applying multicast communication in an existing architecture based on a publish/subscribe message pattern, and how the solution may affect the message handling in the system. The system shall provide message delivery to a group of receivers in a single transmission and allow for communication without intermediate forwarding. The purpose of the work is to investigate the potential of multicast, as a substitute to unicast in a self-protection system for military aircrafts, by the development and evaluation of a proof-of-concept prototype.

1.4 Goal

The goal of the project is to provide an alternative technique for inter-process communication in an existing system, which can decrease redundant traffic in the network and reduce the workload on the sender side. In this way, the time
for distributing messages can be reduced and a more efficient method for data transmission can be created, preparing the system for increasing amount of data which may incur over time.

1.4.1 Benefits, Ethics and Sustainability

With a customized method for transmitting identical information to multiple receivers, it is possible to more efficiently send data over the network with less probability of errors in the message replication process. Potentially, this will lead to decreased time for message handling, and thereby contribute in the development of efficient self-protection systems. Furthermore, this thesis highlights the importance of choosing an appropriate method for the specific needs of a system, and will hopefully benefit others in the process of adapting communication networks of the same nature and with similar quality attributes.

From the perspective of sustainability, consequences and benefits of the current development area can be identified when it comes to ecological and economical sustainability. When considering the aspect of ecological sustainability, where a balanced use of natural resources is desirable, transmission of data over the network is of interest. By reducing the amount of redundant messages over some links, the network usage can be decreased which in turn could lead to less congestion. This will potentially have the benefit of reduced energy consumption and thus a less environmental impact. Moreover, the development of suitable strategies for efficient message handling could eventually reduce the need of more powerful hardware, with higher power consumption, to meet the requirements of the service.

Besides the advantages from an ecological point of view, the introduction of a more standardized technique for point-to-multipoint communication, as the one proposed in this thesis, could contribute in the sense of maintainability. With standardized solutions the possibility of reuse may increase as well, which could be considered as more sustainable from an economical perspective.

Due to the domain, in which the study is conducted, ethical aspects concerning the industry in its entirety always leads to discussions and disagreements. However, when it comes to this project and software development in the current field, presentation of information can be identified as the main ethical aspect. Since the degree project covers development within the area of military defence, publication of sensitive information must be avoided related to the company’s internal data as well as potentially confidential data from a military point of view. This project is therefore carried out in corporation with a stakeholder, where concerned individuals participate in the review of the material before publication.

1.5 Research methodology

To successfully conduct a research project the choice of suitable methods and methodologies are essential, as it affects the outcome and assures quality of the result [11]. Multiple methodologies are available which defines principles and
approaches to plan and carry out the research, and needs to be established in an early stage of a project.

The two most basic categories of research methods usually mentioned are qualitative and quantitative methods [12]. These methods characterize the project, defines a first scientific standpoint and will affect the choice of methods for data collection and analysis [11]. With a quantitative strategy, the study is based on numerical data which is used to prove or falsify an already established theory or hypothesis [13]. The data analysis is then based on statistical methods to explain and present the result of the research.

In contrast to quantitative methods, qualitative research is based on semi-structured data which is used to generate new theories in the current field [12]. By observations and dialogue with involved parties, a deeper understanding of the studied area can be created (but with less accuracy) which is not possible in quantitative research [13]. Qualitative research can thus be described as the use of investigations or development in an interpretative manner to create theories, whereas quantitative methods uses experiments and large data sets to reach conclusions [11].

This degree project aims to address an existing problem in a specific context and generate new knowledge by the development and evaluation of a prototype. For this purpose, a qualitative approach is most appropriate as the development process is based on data derived from observations, discussions and existing knowledge in the current field, rather than numerical data. However, a quantitative comparison between the existing system and the implemented prototype is used as a part of the evaluation. In this way, the result can be validated and a more comprehensive analysis can be provided, indicating whether the concept of multicasting is a suitable solution to the addressed problem or not.

Furthermore, a philosophical assumption is established in order to define the point of view for the project [11]. These perspectives describe the assumptions of what is considered as valid research, where the most common for qualitative research is defined by M. Mayers et al.[14] as; interpretivism, criticalism and positivism. From an interpretative point of view, the reality is assumed to be accessed only through social constructions [11]. The understanding of a phenomena is hence established by explorations of opinions and experiences to identify context for the phenomenon. Criticalism is instead based on the assumption that the social reality is historically constituted and focus on the conflicts and contradictions within the studied area [14].

In general, positivism assumes that reality is objectively given and not dependent on the researcher and instruments, with focus on testing theories [11], [14]. Moreover, A. Håkansson describes the post-positivist assumption as positivism that leans towards interpretivism [11]. With a post-positivist assumption, the reality is assumed to be objective but at the same time takes into consideration that the experiences of the researches may affect the
outcome of the research. As this research is based on observations as well as knowledge gathered from the experiences of individuals, a post-positivist assumption is chosen due to its hybrid nature of positivism and interpretivism.

![Diagram of research approach]

Figure 2: i) Inductive research approach ii) Deductive research approach [15]

Finally, when drawing conclusions from the conducted research, inductive and deductive research approaches are commonly used [13]. The deductive approach aims to prove theories by testing and is often related to quantitative research, whereas an inductive approach is based on a reversed process [12]. An inductive approach is thus based on initial data collection from which knowledge are derived to create theories, see Figure 2, and are usually associated to qualitative methods [15]. Since this project presents an investigation, where the implementation of a prototype is based on data gathered from observations of an existing system, the project applies an inductive approach.

1.6 Stakeholder

With many years of experience in military defence and civil security, Saab offers world-leading products and services for airborne, land based and marine platforms [16]. Self-protection systems for airborne platforms is one category in the context of EW systems which the company provides, to protect pilots from potential threats [17]. With the use of Radar Warning Receivers (RWR) and Electronic Counter Measures (ECM) the purpose is to avoid detection and identification by confusing a potential attacker which may have discovered the vehicle [18]. Through collection of sensor data, the system can provide the user with information, necessary to support decision making. Timeliness, meaning that relevant data is available when needed, is thus an important factor in this context.

To meet the requirements that comes with a constant technical development and changes in the military landscape, Saab is now in the final stage of developing the next generation of EW self-protection systems for the Gripen E/F aircraft [19]. For this purpose, an upgraded version of the communication strategy is of interest.

The existing system is based on a distributed network of different computational devices which are placed on different locations in the aircraft, and collaborates by exchanging information through message passing. As a part of the development process, the department of EW systems is now investigating new strategies for more efficient message handling. This degree project
constitutes one phase in the design process of alternative ways for information exchange in a self-protection system of distributed nature.

1.7 Delimitations
This thesis covers multipoint communication in a self-protection system for airborne platforms, and the potential impact it may have on the communication network. Eventual security aspects, such as encrypted communication, is not considered due to the closed nature of the examined system.

Coordinating actions within distributed systems may be crucial to efficiently solve an addressed problem. The success of coordination is dependent on several factors, where time for message delivery and ordering of messages poses two important aspects. This thesis focusses on IPC in a distributed system with the purpose of reducing time for message handling and network load, and will not consider ordering of messages. More specifically, the implementation will not consider in what order messages are delivered and that they are delivered in the same order by all receivers.

1.8 Outline
This thesis starts off by presenting an extended background with focus on point-to-multipoint communication, where the concept of multicasting and related technologies are explained, which constitutes Chapter 2. Furthermore, the second chapter describes related work in the current field identifying aspects relevant for this project. The method for conducting the project is then presented in Chapter 3, including explanation and motivation for the chosen research strategy as well as methods for data collection and analysis. In the same chapter a presentation of the working process is provided which describes the different activities underlying the result of the project.

Chapter 4 presents the result of a conducted pre-study. To identify the problem situation, and important aspect concerning system development in the current area, an investigation of the stakeholder’s system is performed and presented as a system description. In Chapter 5, the development of the prototype implementation is described together with a presentation of the final communication design. This is followed by an evaluation of the implementation in Chapter 6, where the prototype developed in this project is compared to the original design in terms of performance. This evaluation then forms the basis of the conclusions presented in Chapter 7.
2 Point-to-multipoint communication in distributed systems

Due to the absence of primitives based on shared memory for communication within distributed systems [1], efficient exchange of information through message passing becomes even more important. Based on low-level message passing, offered by the underlying network, interaction between different parts of the system can be provided which may be necessary for enabling coordination between entities.

Today, multiple techniques and protocols are available to allow for information exchange between two endpoints in a system. Applications where the same information is consumed by many is common practice nowadays and applies different types of strategies to achieve point-to-multipoint communication. However, as message passing is the only way for communicating in these systems the load balancing becomes important. This chapter gives an overview of the concept for data communication and a description of commonly used strategies for providing group communication.

2.1 Concept of data communication

Considering the fundamentals of message delivery and packet forwarding, the OSI model [20] can provide a general overview of the different layers used when describing computer communication, see Figure 3. It is a conceptual model used to categorize network protocols and techniques where each layer provides a specific service, independent from other layers.

![Figure 3: The OSI-model](image-url)
Encapsulation is the method for adding information on each layer in the OSI model and describes the flow of data from application level to the physical layer. Moreover, two important concepts defining the mode of data transfer is connection-oriented and connectionless communication [21]. These concepts are fundamentals for process-to-process communication and specifies which protocols could be used for data transmission. In connection-oriented communication, transmission of data can begin first after a connection is established between sender and receiver. With a connectionless communication there is no connection establishment phase, and data is sent directly to the receiver with no delivery and correctness quarantines.

2.1.1 Transport protocols
Data from the application, presentation and session layers is encapsulated at the transport layer together with transport protocol specific information. Through the transport layer, different protocols are available where the Transport Control Protocol (TCP) and the User Datagram Protocol (UDP) are among the most frequently used [22].

UDP is a protocol based on connectionless communication and was designed to send data with minimal protocol overhead [23]. By the use of port numbers, the UDP protocol provides point-to-point communication for the above layers. The protocol does not have support for features like flow control and acknowledgement monitoring and is, in many cases, referred to as an unreliable transport protocol. It is commonly used in applications where fast communication is vital and where the flow control is of less importance, such as multimedia applications.

TCP is an established protocol for connection-oriented data transmission. Unlike UDP, the TCP protocol offers flow control and retransmission of lost packets [22]. Furthermore, it provides features like error control and acknowledgement monitoring. TCP enables point-to-point communication in the same way as UDP, using source and destination port numbers, but requires an established connection which must be open during the whole session. As TCP can offer retransmission of lost packets, due to the acknowledgement mechanism, delivery of messages can be guaranteed making it a more reliable transport protocol than UDP [23]. However, all these features for reliability comes with an extended overhead for each segment sent, and in a situation where frequent retransmission of packets is needed the latencies and high network load will be the result.

2.1.2 Network and Data Link layer
The network and data link layer are the two succeeding levels of abstractions after the transport layer. At these layers the routing process is defined, i.e. the selection of the most suitable path for data transfer between sender and receiver [23]. The network layer is responsible for several functionalities such as network connection, segmentation, error detection and mapping between network addresses and data link addresses [21]. Network connections are provided between transport-layer entities using the data-link layer connections
which are currently available. The segmentation process is then used to split data into smaller blocks to support data transmission between networks with different link layer standards.

To provide its functions, network addressing is an important facility at the network layer. Network addressing refers to the process of giving each host in the network a unique address which is both consistent and required to enable communication. At the data link layer, procedures for interacting with network hardware and transmission medium access are defined [23]. Moreover, the mapping to Medium Access Control (MAC) addresses [24] is performed at this stage.

The leading protocol in this context is the Internet Protocol (IP) [25] which is a connectionless protocol with no acknowledgement mechanism. Hence, the functionality to assure message delivery is not provided through IP and is left out for higher layer protocols to handle. It is a protocol originally designed to work in interconnected systems based on packet-switched communication networks [23]. By adding a header to the data packet, with a source and destination IP-address, the protocol can support message delivery from host-to-host, either sharing the same or being part of different networks.

**2.2 Publish/Subscribe communication pattern**

Sharing information among different applications in a distributed system is essential for coordinating actions, and may be the key to success. However, when sending information among different machines, most of the events are not relevant to all nodes in the system. Hence, distributing all events to all available receivers is inefficient. The publish/subscribe pattern [26] is a widely used strategy for message delivery where nodes register their interest in messages of a certain type. The technique defines a relationship between senders and receivers, where nodes are connected in an indirect manner, to enable group communication.

With the use of publish/subscribe patterns, applications can publish information which is then consumed by subscribers. Furthermore, the subscription model is used to classify the publish/subscribe system into three different categories which is; topic-based, content-based and type-based [27]. In a topic-based system, events are published under a certain topic or subject, for which subscribers express their interest. All subscribers will then receive all messages published on that topic. Content-based systems enables restriction on the event content meaning that messages are delivered to subscribers only if the content matches the pre-defined constraints. Finally, in the type-base publish/subscribe category, events are classified as objects belonging to a specific type. These objects can then encapsulate attributes but also different types of methods.

**2.3 Multicast, unicast and broadcast**

In many existing real-time applications, the ability to deliver data from one sender to multiple receivers is required, like in streaming applications [6]. As
the publish/subscribe pattern provides the functionality of only sending information to interested parties, it does not necessarily specify how data is delivered. Specifications for data transmission over the network is instead defined at a lower level. In this context, three standardized techniques are usually mentioned which are: broadcast, unicast and multicast [8].

The term broadcast refers to the delivery service where packets are transmitted to everyone in the network [28]. This means that all packets will be received by all nodes in the systems, who becomes responsible for the filtering of messages. It is an approach for message distribution normally used in Local Area Networks (LAN) where each frame is sent across the entire network and where, for instance, the MAC address at the receiver can be used to eventually discard irrelevant messages [8].

Unicast is the concept of one-to-one delivery of messages and is the opposite of broadcast [6]. Every packet sent is thus provided with a unique network address to support delivery to a single receiver. Moreover, it is the only approach of the three mentioned that provides message delivery supported by the TCP protocol.

As the third technique for message distribution, multicast [28] provides data transmission which is more restrained compared to broadcast. At the same time multicast offers the functionality of simultaneous delivery of identical packets to a group of receivers, which is not possible with unicast. It could be considered as selective broadcast where data is to be transmitted to more than one receiver, but not to everyone.

Figure 4: Different types of message delivery; unicast, broadcast and multicast. Adapted from [6]

In the context of point-to-multipoint communication, all three principles of broadcast, unicast and multicast can provide this functionality. When using the technique of unicast, the sender simply creates one copy of the message for each receiver and sends it separately over the network. This will however lead to redundant traffic as the same message will be transmitted over the same link multiple times. Even though unicast is supported by the TCP protocol and thus provide a more reliable data transmission, it is considered by several [29], [30] as inefficient when identical messages shall be delivered to many recipients.
With the use of broadcasting as an approach for group communication, all packets will be sent to every node. The replication process will not be handled by software on the sender side which means that the redundancy of sending the same message multiple times on the first outgoing link is reduced, see Figure 4. However, as messages are sent to all destinations without any exception, some nodes may receive messages they are not interested in, leading to unnecessary load in the network. The motivation for multicast in this context is that it offers multipoint communication in a single transmission but only to a specified group. Using multicast, the source could send one single copy of the message to a multicast address, which defines the multicast group. The copying process is then handled by network elements which routes the information to every recipient in that particular group, leading to less redundancy.

2.4 Multicast addressing

To support multicast applications the addressing is an important aspect. Unlike unicast addresses, pointing to a single destination, and broadcast addresses which points to all available hosts, a multicast address needs to define a subset of hosts in the network. For this purpose, the IPv4 protocol provides group addresses for representing a set of hosts. These addresses are class-D IP addresses in the range from 224.0.0.0 to 239.255.255.255 [31]. Each address can be used to define a multicast group for which hosts can become a member of in order to take part of the same information. The functions of a multicast address can be described as [28]:

- uniquely identify a group
- enable multicast routing to build a broadcast tree
- enable routers to forward packets

Furthermore, to be able to reach an IPv4 multicast group over Ethernet [32] the 23 lowest bits are mapped to a MAC-address, creating a MAC-address range from 01-00-5e-00-00-00 to 01-00-5e-7f-ff-ff for all possible multicast groups [31]. Upon sending a message the device driver simply converts the IP address into a MAC-address before distributing it out on the network.

2.5 Multicast routing

Communication using multicast relies on additional functionality in the network to provide correct routing of messages. Since multicast addresses do not specify a unique destination, network elements need additional information to enable message forwarding.

To provide network elements, i.e. routers and switches, with proper routing information hosts express their interest in different multicast groups using e.g. the Internet Group Management Protocol (IGMP) [33]. To indicate on what interfaces a router has members of a specific multicast group, hosts inform routers by sending a signal, thus enabling correct delivery of incoming traffic. IGMP operates at the network layer, however, at the link layer IGMP snooping [28] may be used by a switch to listen on IGMP traffic and creating a map of
links interested in certain IP-multicast traffic. Since switches by default flood all incoming multicast traffic out on all available ports, snooping can be used to prevent undesired traffic on those links with no interest in the current packet.

From the obtained information about group membership, different routing protocols can be applied to build a multicast forwarding tree for efficient routing in a network [31]. As packets may be able to take several ways in a network of distributed nature to reach is destination, a routing protocol for finding the most suitable path is desired. However, due to the decentralized architectural design of the network examined in this thesis, where data only can take one path to reach is destination, routing protocols will not be considered during implementation.

2.6 Related work
Multicast have been a subject for active research during long time, where efficient routing protocols is the major focus for recent publications. However, when considering previous research within the scope of this project, H. Liu et al. [34] presents a study where multicast is compared to unicast in the context of online auctions. The authors highlight the importance of small delays in online auctions in order to provide quality of service for auctioneers and bidders. At the same time, they establish that the use of unicast in this context may cause unpredictable delays and severe waste of network bandwidth when the number of participants and number of items grow.

The proposed solution is a model for multicast-based online auctions, containing the four steps; registration, opening, bidding and closing. During registration the auctioneer and all registered bidders forms a multicast group. The opening step includes the distribution of an initial bid which is sent via multicast to all bidders. This bid may be answered by some bidder which send a unicast message back to the auctioneer in the third step. This bid is then forwarded to all other bidders participating in the auction. After a certain time, the auction is closed and a winner is determined by the auctioneer which is the final step in the model.

From the proposed model, an implementation is developed which is then evaluated in terms of packet delay, traffic rate and packet loss. The result shows that the multicast implementation performs better in all three test cases compared to unicast and that the multicast solution only suffers from a minor packet delay even in a noisy network. The study points out important steps in the development process such as the registration of multicast groups. However, it fails to give a more detailed description of how the underlying communication is provided and how the registration of groups is managed. The result of the evaluation clearly shows that the multicast implementation successfully can replace unicast in the current context. Nevertheless, the study is not conducted in an embedded environment where devices participating in the multicast communication may have limited resources. For the purpose and context of this project it would be desirable to also measure how the workload on the sender node is affected by the introduction of multicast.
Another related study for this thesis project is the one conducted by L. Opyrchal et al. [35] which exploits the use of multicast communication in content-based publish/subscribe systems. The study addresses the problem of creating a suitable set of multicast groups, in a scenario where subscribers require very specific information which may not be required by anyone else. Utilizing a straight-forward mapping between information and multicast groups would thus rapidly grow beyond practical limits. Furthermore, they point out that even though the IPv4 protocol can provide a group address range which is of significant size, the practical limit is smaller since the routing table space in routers is a limited resource.

L. Opyrchal et al. [35] highlights an important aspect which needs to be considered during implementation of multicast in a system. How to divide information into appropriate multicast groups may differ depending on the type of publish/subscribe pattern used. It is, however, important to consider the mapping of information in order to create a suitable set of multicast groups, but as far as possible minimize the delivery of messages to those which are not interested in the message.
3 Methodology

To be able to create a structured research process, an appropriate research strategy is necessary as it defines guidelines for data collection and analysis throughout the project. This project is based mainly on a qualitative approach, and the main question is formulated without further assumptions, which underlies the choice of methods for data collection and research strategy. The process for conducting the project is based primarily on an exploration phase, a development phase and an evaluation phase where data is collected in several steps and with different strategies. These three phases end up in a prototype which aims to serve as a basis for the discussion about the performed research.

Multiple research strategies are available and is applied in a project to organize the research process, where grounded theory, case study and action research are among the most frequently used in qualitative research [11]. Grounded theory seeks to construct theories by continuous collection and analysis of data, but does not have a specified theory as starting point for the project [15]. Case studies and action research is instead based on investigation of a phenomenon in a real life context [11] [36], where case studies could be considered as observational meanwhile action research focus on the change process [15].

As this project focuses on the integration of theory and practice, to address issues in a specific context and contribute to change and practical knowledge, action research is considered as a suitable research strategy. It is a systematic and cyclic method of problem identification, planning, actions, evaluation and reflection, with emphasis on problem-solving in a practical context [11], see Figure 5. Furthermore, action research provides a flexible design which enables adaption of data collection as the project progresses [15].

![Figure 5: Brief description of the different steps included in action research, adapted from [36]](image)

From the action research model presented in Figure 5, a working process is established which, in more detail, specifies the different phases and activities in this project. The exploration phase consists of observations of the stakeholder's
system and problem identification, while the development and evaluation phase include activities for realizing and validating the implementation of the prototype. Through this process, data is collected in several steps to support the implementation and evaluation. The remaining parts of this chapter details the methods used for data collection and analysis in this degree project and how they are applied in different stages of the project.

3.1 Data collection methods
The process for data collection constitutes a significant part of the research and is a continuously ongoing activity during the project. This includes a number of elements which together forms the basis of the development and evaluation phase. By using different methods for data collection, when studying the same phenomenon, the advantages of independent confirmation can be achieved which is considered as valuable in this context.

A literature study forms the theoretical background of the report, where data is collected to identify important concepts and theories in the studied area. Furthermore, a pre-study is performed to understand how the current system is implemented and limitations of the communication strategy used by the stakeholder today. Through documentation, source code and interviews, data is collected in order to create an overview of the system design where important aspects for the development phase is identified. The evaluation of the implemented prototype is then based on experiments where data is collected to validate its potential.

3.1.1 Documents study
In qualitative studies such as action research, knowledge about history and background is often required in order to understand the current situation and problems underlying the desire for change [37]. For this purpose, data can be collected with different methods and from multiple sources where document analysis constitutes one of the most common strategies [38].

Studying documents and archival data may include different types of available material, independently analyzed by the researcher. For example, data can be collected with the use of meeting minutes and management documents or from more technical documents and source code [15]. For this project, knowledge and data is gathered by studying and analyzing source code and available documentation concerning the examined system. In this way, a proper description of the system can be created which is used later in the project.

3.1.2 Interviews
Data collection through interviews is a frequently used method within several research disciplines, and often constitute an important source of information. The reason for this is that individuals working within the field of research in many cases possess knowledge that cannot be obtained only by studying documentation [15]. Interviews as a method for data collection in scientific research includes several categories, and can be used as either a primary data source or as a validation strategy, depending on the project’s nature. Usually,
three types of interviews are mentioned in this context, characterizing the procedure, which is unstructured, semi-structured and structured interviews [12].

In a structured interview, the dialogue between researcher and interviewees is based on a pre-defined question scheme, normally including detailed and closed questions with less room for vague definitions [39]. This technique is often used in quantitative research as it enables comparison between participants and simplifies the analysis. When it comes to semi-structured and unstructured interviews, questions are formulated in a more open way [15]. The interview process tries to capture the opinion of the interviewees in order to create a broader understanding of the studied area, which is feasible from an exploratory point of view. These strategies share many attributes, but what separates them is that semi-structured interviews are based on a guide of pre-defined topics whereas unstructured interviews use the guide more as memory aid [12]. However, in both cases the interview may be influenced by the progress of the conversation.

This degree project applies an unstructured interview technique where data is collected in addition to the knowledge gathered from the documents study. With unstructured interviews there is more room for discussions which provides the opportunity to clarify and validate different aspects, and is considered as important in order to create a clear picture of the studied system. By utilizing the knowledge of experienced people in the field, data can be collected necessary to understand the problem situation.

3.1.3 Experiments

Experiments [11] is yet another method for data collection which can be used in a research project. It is a method where larger amount of data is collected for a set of variables in an attempt to understand cause-and-effect relationships. With the use of experiments, quantitative data is collected, hence, it is more common in investigations of quantitative nature.

Although experiments on its own might not provide a deeper understanding of the studied phenomenon, it can generate results on causal relationships [15], and could therefore be used as a complement to qualitative data collection methods in action research. With the use of experiments, the effects of direct, precise and systematical manipulation of behaviors can be measured, which are useful when changes are evaluated [40]. The process of conducting experiments can be made in an off-line situation, e.g. in a laboratory simulating real world events, or on-line where the investigation take place in a real-life context of the studied field.

Since this project includes the development of a prototype implementation, experiment is considered as a suitable method for evaluation in order to measure the effects of the implemented changes. Moreover, by collecting numerical data, it enables the comparison between the prototype implemented during this project and the current system used by the stakeholder. Due to the
fact that the system, in its original context, is used in military aircrafts an off-
line situation for simulating real world events is chosen for the experiments.

3.2 Data analysis
To analyze the collected material, different methods can be used describing
guidelines for how to create a structured data analysis. This includes the process
of inspecting, selecting/filtering, transformation and modeling of data, and are
used to support decision-making throughout the research process [11].

In qualitative research, data analysis focus on extracting patterns from concepts
and creating insights whereas analysis of quantitative data is used to support
conclusions. Two commonly used methods for qualitative data analysis are
coding and grounded theory. Coding refers to the process of analyzing
transcriptions of e.g. interviews and convert that into quantitative data by
assigning labels and thereby attach meaning to the data [41]. Grounded theory
is an iterative process of collection and analysis with focus on theory discovery.
It allows the researcher to identify general features of a topic and create a
theoretical account [42]. For this degree project, grounded theory is chosen as
an analysis method since data collected about the current system aims to create
understanding and knowledge, later used in the development phase.

As experiments are chosen for evaluation in this project, a discussion about data
analysis for quantitative data is appropriate. In this context, statistics and
computational mathematics are the most common methods [11]. Statistics is
used to analyze data by calculating result for a sample and evaluating
significance, whereas computational mathematics focus on calculation of
numerical methods. For the evaluation of the implemented solution, statistics
is chosen as method for data analysis to properly represent the measurements
that underlies the result of the evaluation.

3.3 Working process
To efficiently steer the project towards its goal, an appropriate working process
is established which defines the framework for organizing the work and
controlling the development process. The fundamentals of action research
presented in Figure 5 have been used to define a transparent process for
conducting this degree project, including a description of the different stages
and their purpose. In Figure 6, an illustration of the working process is
presented.
The flowchart illustrated in Figure 6 defines the workflow for the project, including the set of activities reflecting the exploration-, development- and evaluation phase described earlier. From the described methods for data collection, the procedure of collecting and utilizing the information during the different stages of the project is presented.

### 3.3.1 Pre-study

As an initial phase of a software development project a pre-study may be necessary to clearly identify the current problem situation [43] and facilitate the specification of requirements. This is in accordance with the diagnosis phase of the action research model and is used to generate sufficient material to be able to proceed to the next phase of a project.

With the purpose of understanding the communication mechanism used by the stakeholder today and the system limitations, a pre-study is conducted where different functionalities of the existing system are investigated. This includes the process of studying documentation and source code together with interviews to identify core aspects of the system. Findings during this phase of the project is then utilized to establish requirements for the final design proposed in this thesis.

The analysis of available documentation and source code is an alternating process. Documentation is used to create an overview of system components and different techniques used for communication, and the source code is used to more deeply dig into technical details and algorithms. By observing the system behavior, key fundamentals can be identified which is essential knowledge when constructing the prototype.

In order to get an even deeper understanding of the problem situation and the needs for improvements, interviews are conducted with different individuals both on individual basis and in groups. For these interviews an unstructured
interview technique is applied to steer the conversation more towards a
discussion with focus on confirming knowledge gathered from the documents
study. The interviewees all have different roles at the company and is chosen
due to their expertise in the studied area and knowledge about the current
system design. The three roles can be described as:

- Specialist in software architecture
- Scrum master for the team currently working with the communication
  system
- Software engineer

Notes are taken during the meetings to highlight important factors regarding
the system and its components. Subsequently processing and interpretation of
all the material is made to finally present the result as a system description and
the current limitations.

### 3.3.2 Requirements elicitation

To provide guidelines for the design of a system, requirements must be
established describing goals and expectations for the project [44]. Through the
collection and analysis of requirements important aspects and problems can be
identified, which are useful in the implementation phase of a software
development process.

Specification of requirements constitutes the second step in the working
process of this project, and is conducted in order to support design decisions
during the development phase. The data used for this purpose is collected
during the pre-study and analyzed to extract a set of goals for the
communication design.

### 3.3.3 System development

Different models for system development can be used to identify significant
stages and methodologies of a project. The waterfall model [45] and agile
software development [46] are two of the most common methods for
conducting software development projects and applies different strategies to
steer the work [45]. With the waterfall model, the project is divided into a
number of phases which are performed sequentially and in chronological order.
This means that a new phase only can be started when the previous step is
completed. This creates a structured working process but is not well suited for
handling problems that may arise in the test-phase, since it is not possible to
return to previous phases.

In agile software development, the development process is less sensitive for
changes and allows for iterations of the different phases in a project [46]. Parts
of a system can thus be completed before implementation of the next
requirement is undertaken, and the result of one implementation may therefore
influence the rest of the requirements. Project of this nature may provide
elasticity but makes it more difficult to provide good time estimations.
With respect to the presented models for development projects, this project applies a method with clear agile elements to create flexibility and support for eventual changes during the project. The system development phase in this project is divided into three different activities; design, implementation and testing. These activities are included in an iterative process, see Figure 6, to enable implementation and testing of some functionality before starting with the design of next component. After conducting the pre-study and requirements elicitation the development phase begins, where one iteration includes the following:

**Design** includes the definition of the functionality that should be implemented, and is the initial phase of the iteration. In the design-step a plan for how to test the chosen component is also defined.

**Implementation** is the phase where the functionality is developed. Thanks to the flexible design of the development model, all functionality may not be completed in this phase before moving on to the test phase. Any remaining part are moved to the next iteration.

**Testing** is used to make sure that the implemented functionality fulfils the requirements. The test phase is thus used for verification purposes. Eventually this will lead to insights useful in next iteration, when designing next component.

When all functionality is implemented and tested to verify that the prototype satisfies the requirements, the development phase is considered completed. The result from this phase can then be evaluated to define accomplishments of the project.

### 3.3.4 Evaluation

Measuring the effects of performed activities is the final step in the data collection process of an action research project [36]. This includes evaluation of implemented changes to determine potential achievements, underlying the conclusions of the conducted research.

To evaluate the final implementation of this project, experiments are conducted where numerical data is collected and analysed with quantitative methods. Based on available hardware, measurements are performed to investigate how well the implemented solution in this project perform compared to the original communication design used in the stakeholder’s system. For this purpose, experiments are designed to measure latencies but also how the system behaviour is affected by an increased amount of data and number of clients. Test cases are thus formulated to enable comparison between the two implementations, showing the potential of a changed communication strategy. Results from measurements are then presented with statistical methods where accumulated data sets are used based on several identical iterations.
4 Pre-study of existing system design

For the purpose of identifying the current problem situation, a pre-study of the stakeholder’s system is conducted. By investigating how the system is constructed and what communication strategy is used today, limitations can be identified. Through documentation, source code and interviews, data is collected which is later used to establish requirements for the development phase of the project. For this inquiry a set of topics is considered to provide a general picture of the existing system and drawbacks of the communication design. The following points forms this pre-study:

- Identification of network architecture
- Model used for communication
- How messages are delivered
- Problem situation

The system examined and used as a basis in this degree project is a distributed system of between 10 and 20 subsystems, also called nodes. These nodes represent different computational devices, sometimes holding sensors, and communicates through message passing in order to provide situational awareness for military aircrafts. This chapter summarizes the knowledge gathered from the exploratory phase of the project and presents the result as a system description of the stakeholder’s system. From this, important requirements are acknowledged which are taken under consideration during the implementation of the prototype.

4.1 System overview

Looking at the overall structure of the communication network, it can be described as a decentralized topic-based publish/subscribe architecture [27]. Nodes in the system are divided into pre-defined domains, which symbolizes event groups, for which the included nodes collaborate by sharing information. A domain is defined by the set of events it contains and nodes within a domain are those with an interest in events of a certain type. Events together with domains then forms topics for which nodes can subscribe and publish to.

Every node in the system can have the role as a publisher and/or a subscriber and can be a member of several domains. Some nodes may also act as a bridge between different domains, meaning that they might subscribe for events in one domain and then publish the result in another domain. Besides publishers and subscribers, the system contains one server node. This node holds information about domains together with active publishers and subscribers in order to provide a common view of the system. Since nodes are developed independent from each other, the server can provide them with information to enable communication. This is achieved by having all nodes register for domains and topics at the server node. In Figure 7, the architectural design is illustrated, which gives a theoretical overview of the existing system and its structure.
The network is based on a set of hosts connected by switches and where communication take place over Ethernet\cite{32} to forward information between publishers and subscribers. This is illustrated in Figure 7, which also shows how nodes can be divided into different domains.

With the responsibility of providing the registration service for the remaining parts of the system, the server node serves as dictionary to supply other nodes with knowledge about how messages should be distributed. This node manages registration requests, received by either a publisher or a subscriber, and provides information to enable communication. Additionally, the server has the responsibility of monitoring, which includes continuous identification of available resources in the system.

4.2 Communication model

With a topic-based publish/subscribe model for message distribution, the current system provides direct communication between sender and receivers, without any interference from other nodes in the system. As previously mentioned, the system includes a server node with the responsibility of registration but with no impact on the communication at later stages. After the configuration and registration phase, the server node will thus have a less prominent role as messages will not go through that node to reach its destination.

During configuration, publishers and subscribers provide an internal registration in order to identify what domains and events are of interest. Internally, each node identifies what topics it should publish and subscribe to by reading from pre-defined configuration file. After this process, each node
contacts the server node to register for all domains of which it should be a part of. A successful domain registration will then enable registration for topics as either a publisher or a subscriber, see Figure 8. When a publisher wishes to register for a topic, the server provides the publisher with a list of subscribers for that particular topic. By an internal lookup, the server node identifies which subscribers are registered for a topic and communicates this to the publisher.

Figure 8: Sequence diagram showing the concept for communication in the stakeholder’s system
In the other case, when a subscriber registers for a topic, the server forwards the information to all concerned publishers, keeping their subscriber lists updated. This list is continuously updated as new subscribers register for different topics in the system or when a subscriber is no longer available due to potential failure. The list of subscribers, which publishers receive from the server node, gives the publisher an opportunity to handle the communication without any intermediate forwarding. Messages can thereby be sent directly to subscribers without going through the server.

To monitor the availability of resources in the system the server node also sends out heartbeat messages [47] to verify which nodes are alive. Heartbeats are messages being sent periodically to all nodes in the system to provide failure detection. This implies that nodes which do not respond to heartbeat messages within a predefined time span will be considered as dead, and therefore needs to be re-registered to be included in the network again. The heartbeat functionality is represented as a loop in the sequence diagram of Figure 8, and is a continuously ongoing process providing the server node with information which helps in determining the state of the system.

4.3 Message delivery

From the described communication model, it is clear that the information exchange between publisher and subscribers is provided in a peer-to-peer manner. When a publisher is supposed to distribute information, it iterates through its list of subscribers provided by the server and send data out on the network which is redirected to the correct destination with the use of network switches.

The underlying message delivery technique is based on the UDP protocol which provides a stateless communication between nodes in the network. This means that no acknowledge monitoring is available, and retransmission is handled at application-level. The ACK’s represented in the sequence diagram of Figure 8 is hence regular messages sent from the server but with the same purpose as ACK-responses. Furthermore, nodes apply unicast when delivering messages meaning that one copy of an event is sent to every subscriber registered for a specific topic. The copying procedure is thus provided by the software on each sender, and is delivered one by one to recipients in the system.

4.4 System limitations

Through examination of the stakeholder’s system, it can be determined that the currently used strategy for inter-process communication provides direct message delivery between publishers and subscribers where events are distributed with the use of unicast. However, when using unicast for delivering events, the publisher becomes responsible for the replication process which can be considered as a limitation during the information exchange. Instead of having the server node as an intermediate forwarder of messages, leading to one message going out from a publisher, the peer-to-peer strategy may cause high workload on sender side. Since publisher nodes have less capacity than the server, this may lead to latencies when the node should provide messages to
many subscribers. As publishers must send one instance of a message for each subscriber interested in a specific topic, the processing time for message handling will be affected negatively.

Generally, with unicast as a communication strategy redundant traffic on the outgoing link can become a problem in a system with data-intensive communication. Moreover, the potential risk of incorrect copies is also present when the replication is controlled by software. In a system design where a central unit is responsible for the registration of publishers and subscribers the problem of single point of failure may arise. The current system is thus limited by the fact that if the server node goes down during the registration phase the communication between publishers and subscribers may not be possible or incorrect if they have not received all information from the server.

From the conducted pre-study it can be established that the currently used communication strategy is not fully adapted to handle large amount of data exchange between entities in the system. The problem situation can thus be identified as insufficient use of available resources in the system due to redundant traffic in the network and undesired load on sender side. To improve the communication design, the replication process should be moved out from the nodes in order to reduce the processing time and minimize potential faults.
5 Applying multicast communication in a data-intensive distributed system

In this chapter, the process of applying multicast in a data-intensive distributed system is presented. This is done by replacing an existing communication strategy based on unicast with multicast, for the system provided by the stakeholder. Using the system design described in Chapter 4 as a basis, this chapter details the implementation and final design of the prototype developed in this project.

5.1 System requirements

To ensure that the development process is conducted in accordance with the purpose and objectives of the project, system requirements are defined providing goals for the final product. Since this degree project is carried out on the behalf of Saab [48], it is primarily their interests which underlie the specified requirements for the prototype. These requirements form the basis for the design of the implementation and are extracted from the conducted pres-study, presented in Chapter 4.

The pre-study describes the general concept for data communication in the stakeholder’s system and the current problem situation. By identifying limitations of the existing system, through interviews and discussions with concerned individuals at the company, important functionalities for the final implementation can be highlighted. From the data collection process of the pre-study, the following requirements have been established:

- **System functionality**
  The implementation should not change the original functionality of the system.

- **Low-level acknowledgment**
  The communication should not use an underlying protocol based on ACK’s, i.e. no low-level acknowledgement should be used.

- **Multicast**
  The implementation shall be able to provide multicast communication between publishers and subscribers, and thereby replace the existing communication based on unicast.

- **Subscribers**
  Subscribers should be able to register for several multicast groups.

- **Publishers**
  Publishers should not need to know what topics subscribers are interested in.

- **Message forwarding**
  The distribution of messages to each individual node should be managed by network devices.
**Message delivery** Messages should not be delivered to nodes with no interest in the current domain.

Although the requirements are specified based on the desired behavior of the examined system, they represent underlying properties which is of great importance in software development within similar areas. From the initial requirements and the knowledge derived thus far, a set of non-functional requirements is identified which are:

- **Testability** The implementation should not aggravate the possibilities of testing in the system.
- **Observability/monitorability** It should still be possible to monitor all traffic in the system.
- **Timeliness** The communication should have a reasonable short delay time.
- **Performance** The implementation should reduce time for message handling and load on sender side.

The above-mentioned criteria specify quality attributes for the system rather than specific functionalities, but have an important role when it comes to the design proposed in this thesis. Testability can be described as the extent to which feasible tests can be designed in order to determine whether a requirement is met [49]. A high degree of testability is important within industry since testing is an extensive process to be able to ensure correctness before delivery. Moreover, observability, meaning the ability to monitor activity in the system is yet another important aspect to enable insight during testing.

The aspect of timeliness could be directly connected to the requirement about low-level acknowledgment. Since systems within military defense are time-critical, low latency and timeliness is essential meaning that relevant data should be available when needed. Avoiding protocols based on ACK's is therefore desirable, to minimize the delay, and should be counted for during design activities. Finally, the performance requirement highlights the underlying desire for change in the original system. Together, these describe the expectations on the final implementation from the product owner and forms the guidelines for design choices made throughout the project.

### 5.2 System implementation

Based on the previously presented system development model, the process of developing the prototype includes a number of iterations of the design-, implementation- and test phase. In accordance with an agile working process, the implementation is divided into several steps where different parts of the system are constructed independently and later combined to a final design. By analyzing the system requirements, the design of system components can be
established. This section describes core components and the different functionalities implemented in this project to enable multicast communication in the examined system.

5.2.1 Overview
With the original functionality of the stakeholder’s system as a basis, the communication architecture is extended with a set of features to provide point-to-multipoint message delivery in a single transmission. This means that the publish/subscribe pattern still have a significant role in the multicast implementation and extensions are implemented with this pattern as a basis.

The process of enabling multicast communication could be divided into two major steps. The first step concerns the implementation of logic to handle group communication based on multicast addresses on both sender and receiver side. This includes both multicast addressing and the creation of multicast groups. It also covers the programing of network sockets to support multicast traffic to and from each host in the system. Furthermore, the functionality of group membership registration is established to be able to receive data from certain topics.

Besides the adaption of the communication layer at each publish/subscribe node, the second step in the development process concerns the configuration of network elements to provide correct multicast routing. This requires extension of the original logic provided by switches in the network to support forwarding of messages labeled with a group address. For this purpose, a configuration script is implemented to establish an address table for each switch in the system. Important to highlight is also that the sever logic is not extended with any functionality and the external registration phase illustrated in Figure 8 is kept unchanged. The development is instead concentrated around the enabling of multicast communication between publishers and subscribers and the functionality needed for this purpose.

5.2.2 Data distribution service
To provide a flexible system design, modularity through separation of functionalities is an important aspect. Each node in the system may include a set of services which are requesting or distributing information to and from other parts of the system. This requires functionality for external communication, but instead of having each service responsible for their own network communication this functionality can be extracted into a separate component. By not exposing the communication strategy for different services, i.e. publishers and subscribers, enables the possibility of replacement which makes the design more modular.

For the purpose of managing the inter process communication, a service for distributing data is available at each node of the original system. This component provides an interface between services and itself, but also between itself and the underlying socket communication logic. The data distribution service act as the intermediary for message handling at the node. An illustration
of the concept for data flow can be seen in Figure 9. Incoming traffic passes through a listening socket which the data distributor reads data from. This data is then processed and distributed to services with an expressed interest in the current topic. To handle outgoing traffic the reversed process is performed.

To maintain modularity, majority of the functionality implemented on sender and receiver side in this project is placed in the data distributor and its related interfaces. The implemented logic is hence not exposed to publishers and subscribers which send and receive messages in the exact same way as in the unicast version of the system.

5.2.3 Multicast groups and addressing
An initial step in the implementation process is the creation of multicast groups. This includes the mapping between information and multicast IP-addresses to categorize and determine what groups certain information should be distributed to. A procedure for dividing information into appropriate classifications and associate them with a specific address is thus necessary.

In a topic-based publish/subscribe architecture the division of information into multicast groups may be done on different levels. By utilizing the fact that events in the examined system are associated with a domain, the creation of multicast groups could be made at domain-level. This means that each available domain becomes associated with a group address, which was desired by the stakeholder.
The functionality of multicast group creation and addressing is integrated in the internal registration phase of the node, see Figure 8. During the internal configuration, domains and publishers/subscribers are registered in the data distributor to enable correct distribution of incoming and outgoing traffic. Each service at the node is aware of their own interest in certain topics, through pre-defined configuration files, and sends a registration request to the data distributor in order to register itself as a publisher and/or subscriber for different events.

The registration phase is further extended with a mapping between domains and multicast addresses in the range from 224.0.0.0 to 239.255.255.255. In this way, the data distribution service can maintain domain objects containing associated group addresses and events of the domain. This enable both forwarding of received data to subscribing services as well as publication of events to a group of receivers.

When a publisher issues to send an event, the data distributor performs an internal lookup of the domain of which the event is a part of and the address related to the current domain. With the use of a group address the publication of messages can be achieved in a single transmission and the list of subscribers provided by the server node is no longer necessary. The new functionality gives the data distribution service the possibility of sending a single instance of each event and eliminates the procedure of iterating through all available subscribers interested in the current event.

5.2.4 Communication sockets

One of the fundamentals for enabling communication between two entities in a network is the establishment of sockets. It defines an internal endpoint within the host and is used by a running process in order to send and receive data to and from external sources.

After definition of multicast groups, for classification of information, nodes are prepared for external communication by programming of network sockets. The classification of sockets can be related to the transport protocol (TCP or UDP) used for data transfer and defines different properties for the socket. Since TCP-sockets does not have support for multicast, UDP-sockets is the given choice for providing communication between publishers and subscribers in the system. This is also in accordance with the requirement about non-use of protocols based on low-level acknowledgements. In the original system, information transmission between entities is provided with the use of UDP-sockets. However, as the existing system uses unicast for message delivery the sockets need to be reprogrammed in order to handle multicast communication.

Upon configuration each node creates a UDP-socket which is referred to by the data distributor, through a socket handling object, to receive and transmit data over the network, see Figure 9. As the communication is connection-less there is no need for establishing a connection between publishers and subscribers to allow for communication. In order to properly receive messages, sockets at the
receiver side is bound to a specific predefined port number common for all subscribers to listen on. At the sender side the binding operation is not required since eventual responses can be sent using the source information of the initial message. An illustration of the socket communication can be seen in Figure 10.

When a publisher is supposed to transfer information to a set of subscribers, the data distributor uses the socket handling object to interact with the socket. By providing the socket handling object with the multicast address related to the current event and the subscriber port the socket provides distribution of information. To enable this functionality, the socket options is adapted to handle IP addresses of multicast type and a proper local network interface is determined which supports multicast traffic.

5.2.5 Group membership

Applying multicast in a system also includes the process of joining subscribers into different multicast groups. The group membership registration is done in order for subscribers to receive messages labeled with a group IP-address and only receive events that are of interest. This functionality is implemented by extending the registration phase at the data distributor.

When a service register itself as a subscriber on a topic at the data distributor, it configures the socket by creating a structure holding information about the multicast group which is then associated with the socket by setting the socket option IP_ADD_MEMBERSHIP [50]. The values stored in this structure is the multicast address, obtained from the domain information, and a network interface to listen for incoming traffic on. This process is repeated for every service registering itself as a subscriber for a specific topic. In this way, services can obtain messages from several multicast groups which was identified as a requirement for the final design.
5.2.6 Switch filtering

So far, the implementation needed on software nodes in the system, i.e. on sender and receiver side, has been presented. However, the introduction of multicast requires adaption of logic on network elements in order to route messages to correct destinations. A network environment built upon switches must provide some kind of filtering mechanism for incoming packets to utilize the potential of multicast.

As packets are no longer labelled with IP-addresses to individual nodes, switches in the network needs to be extended with additional information to be able to filter and route messages to all interested hosts based on a group address. This is essential as switches by default broadcast all traffic with a multicast destination address on every available outgoing port. To prevent this, all switches must establish a table of MAC-addresses representing multicast groups and the location of hosts which are members of the groups. The system examined in this project is of embedded nature and the address-table for switches needs to be populated during the start-up phase which should be an automated process. This is achieved by the implementation of a script which generates a configuration script for each network device.

Each switch contains a forwarding database (FDB), which holds the MAC-address table containing all known MAC-addresses and which VLAN and ports they are associated with. The concept for this is shown in Figure 11 where the addresses represent multicast groups. As functionalities like IGMP snooping is not available, the configuration is accomplished with the use of a network topology description.

![SW1 MAC Address Table](image)

**Figure 11:** The concept of MAC-address table for multicast routing
To establish the address-table, a Python script is implemented which parses data from a network description and generates configuration commands which is executed through the command-line interface (CLI) [51] provided by the switch. The topology describes structure of the system and includes; domain to multicast group mapping, location of nodes in the system and which nodes that are interested in certain domains. The python script collects the IP multicast addresses and converts them into MAC-addresses. For each MAC-address, the information about interested nodes and what ports they are available on is extracted. Lastly, the MAC-addresses are combined with the port numbers to produce configuration commands which creates entries in the FDB on the format presented in Figure 11.

5.3 Final communication design

The development process presented in section 5.2 has resulted in a communication design which can provide data distribution between a publisher and several subscribers in a single transmission. With the implemented communication strategy, the system can provide message delivery which is less dependent on the capacity of publisher nodes and reduce the number of identical messages on outgoing links.

Instead of having the publisher node sending one instance of an event to every interested subscriber individually, the distribution of messages is now handled by hardware on switches. With the use of multicast as a substitute to unicast, events are labelled with a destination address of IP-multicast type which removes the iteration of subscribers during message delivery. This iteration, provided by software on sender side, is extracted out on network devices which can provide the forwarding almost instantly and with less risk of undesired modification of data. The procedure of creating a working communication design, has included the implementation of additional logic in the configuration phase of the system to finally be able to send and receive data between entities.

5.3.1 Configuration

During start-up of the system, a configuration process is performed as a preparing step to set up the network and define properties for the publish/subscribe communication. This includes the configuration of switch-cores to establish a forwarding table for incoming traffic and is an automated process thanks to the implemented script presented in 5.2.6. With this script the FDB is populated with all MAC-addresses which corresponds to existing multicast groups. Incoming packets can thereafter be redirected to those ports where interested subscribers are located.

Furthermore, the configuration phase includes an internal registration at each node. Start-up of an application implies a set of stages to be able to prepare for inter-process communication. Each application running in the system may provide a set of services which request and send messages to services in other applications. Upon initiation, services register themselves as publishers and/or subscribers for different topics at a data distribution service which is responsible for incoming and outgoing traffic from the node. The data
distribution service maps domains with multicast addresses internally and gradually fills these domains with events and interested subscribers as it receives registration requests from services.

Additionally, the data distributor opens a UDP-socket for receiving and sending data over the network. For this, socket options are set to handle packets with multicast destination addresses. In the case of subscribers, they are added to multicast groups at socket-level. Whenever a socket is opened the node can listen for incoming traffic which is handled by a separate thread. The data distributor uses the listening thread to periodically listen for incoming traffic received by the socket to distribute it to interested services.

5.3.2 Sending and receiving data

When all communication setup is established the system is prepared for information exchange. The communication pipeline for sending and receiving data in the system now includes the step presented in Figure 12.

![Figure 12: Illustration of how data is sent between entities with the final implementation](image)

Once a service publishes on a specific topic it sends a publish request to the data distributor which perform an internal lookup of the multicast address associated with the domain of the topic. The data distribution service creates a message from the information received by the service and send this message once to the socket along with a multicast destination address and a port number. The socket then sends this message out on the network. When the message arrives at the switch, the device search in its internal address table to find what port to use for directing the message. This is achieved by matching the destination address of the incoming packet with an entry in the FDB. As the message arrives at the requesting host the listening thread, handled by the data
distributor, reads data from the socket for further distribution. By extracting the domain and event information from the message, the data distributor can provide a lookup of interested services. Finally, the data is delivered to correct services, which process and act upon the received information.

5.4 Testing and verification

To verify the correctness of the implementation, tests are performed continuously during the development process to confirm that each functionality fulfills its purpose. Furthermore, a verification scenario is set up to test that the implementation provides the sought-after behavior to the system.

Testing individual fragments of the implementation includes the verification of the socket communication, group membership and the switch-configuration script. The functionality of the socket is verified by controlling that events which are sent by one process could be received by another process. To further validate that the group membership functionality is successfully implemented, it is merged with the socket communication test to see that a process receives messages tagged with a multicast address. Finally, the configuration script is tested by controlling that each multicast group is successfully registered in the FDB. This is achieved by running the python script which then executes the configuration commands in the CLI of the switch. By retrieving the data stored in FDB it can be confirmed that the switch is properly configured.

Moreover, to verify that the implementation can provide a correct multicast communication in a network, a scenario is set up including the following:

1. One publisher P1 which publish on one topic t1
2. One publisher P2 which publish on one topic t2
3. One publisher P3 which publish on one topic t3
4. One subscriber S1 which subscribe on t1
5. One subscriber S2 which subscribe on t1 and t2

During the test all publishers send one instance of their topic. The scenario is constructed to validate that the multicast implementation works as intended. The message from P1 should be received by both S1 and S2 and the message sent from P2 should be received only by P2. This shows that the implementation can handle message delivery to several subscribers interested in the same topic and that only subscribers with an expressed interest in a topic receives the data. This indicates that the switch does not broadcast all messages labelled with a multicast address. Messages sent by P3 should not be received by anyone and should instead be dropped by the switch and is yet another scenario for controlling that the switch filtering is correctly implemented.
Table 1: Functional requirements for the implementation and how they are fulfilled

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Fulfillment</th>
</tr>
</thead>
<tbody>
<tr>
<td>System functionality</td>
<td>The system can still deliver data between entities to provide situational awareness, the structure of events containing the information is unchanged.</td>
</tr>
<tr>
<td>Low-level acknowledgment</td>
<td>The implementation provides message delivery based on the UDP transport protocol which does not build upon acknowledgements nor an establishment of a connection between two parties.</td>
</tr>
<tr>
<td>Multicast</td>
<td>Using multicast group addresses and a forwarding table on switches, the implementation can provide multicast communication instead of message delivery based on unicast.</td>
</tr>
<tr>
<td>Subscribers</td>
<td>During configuration services can register itself for all topics of interest and becomes associated with one multicast group for each domain to receive relevant information.</td>
</tr>
<tr>
<td>Publishers</td>
<td>With the use of group addresses the publisher does not need to know which subscribers that are interested is certain topics.</td>
</tr>
<tr>
<td>Message replication</td>
<td>Through a MAC-address table placed in the FDB at each switch, message forwarding can be handled by network elements.</td>
</tr>
<tr>
<td>Message delivery</td>
<td>By performing a lookup in the MAC-address table, incoming packets can be redirected to the correct destination.</td>
</tr>
</tbody>
</table>

When all tests are conducted, both on single components and together, to verify correct behavior of the multicast implementation, the requirements are reviewed to ensure that they are fulfilled. The result is presented in Table 1 including the requirements and a description of how the implementation meet the goals stated in the requirement specification. The non-functional requirements are later discussed in chapter 7.
6 Evaluation

To evaluate the concept of multicast, a set of experimental tests are conducted with focus on performance. Besides the verification tests presented in section 5.4, where correctness of the implementation is determined, measurements are performed to analyze the potential of multicast as a substitute to unicast. The goal is to evaluate how well the implementation proposed in this thesis performs in comparison to the stakeholder’s system. This evaluation together with the implementation forms the result of the project.

6.1 Test environment

Conducting the evaluation implies the setup of an appropriate test environment. Introducing multicast as a communication strategy in a self-protection system for military aircrafts is a process extending over a longer period of time. A more comprehensive evaluation using the test-rig at the stakeholder is therefore left for future work. However, to evaluate the concept and its potential, estimations on several performance attributes are conducted. Based on available hardware, a simulated laboratory environment is constructed to test the implementation against the original system. The environment is built upon three different physical computers, each hosting a set of processes.

6.1.1 Test setup

All experiments are executed in a test-network based on three physical computers, interconnected by a switch and where the communication is provided over Ethernet. These computers are placed inside an isolated VLAN to minimize traffic from other sources which may affect the result. In Figure 13 the test setup used for this project is illustrated.

![Test setup diagram](image-url)
The components needed for simulating the system are distributed on the different computers in the VLAN where each of them have a set of responsibilities. All tests in this evaluation are based on the scenario where there is one server, one publisher and $N$ number of subscribers. Computer C1 hosts the server application as well as the publisher which is responsible for distributing data during the experiments. The second and third computer (c2 & c3) are used to simulate a set of subscribers in the system. C2 and C3 hosts half of the running subscribers, which are all registered for the same topics during the experiments.

6.1.2 Hardware and software
All computers described in the test setup are Linux machines with similar properties. In Table 2 hardware information for the machines used during the experiments is presented.

<table>
<thead>
<tr>
<th>Machine</th>
<th>Memory</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>8GB, 1600 MHz</td>
<td>Intel i3-4130 dual-core, 4 threads @3.40GHz</td>
</tr>
<tr>
<td>C2</td>
<td>8GB, 1333 MHz</td>
<td>Intel i3-2120 dual-core, 4 threads @3.30GHz</td>
</tr>
<tr>
<td>C3</td>
<td>8GB, 1333 MHz</td>
<td>Intel i3-2120 dual-core, 4 threads @3.30GHz</td>
</tr>
</tbody>
</table>

Moreover, two types of analyzing tools are used during the experiments to collect and filter data. To capture incoming and outgoing traffic the packet analyzer tcpdump (version 4.9.8) [52] is used. By sniffing the network connection, all packets sent and received on the machine hosting the publisher can be collected. This is done in order to record timestamps of incoming and outgoing packets for latency tests. Filtering of data is then done with the use of Wireshark (version 1.10.14) [53] to be able to calculate the delay between packets. Lastly, for measuring time within the application a c++ library is used.

6.1.3 Parameters
Construction of test cases for the evaluation is based on a set of parameters which represents different attributes in the system. Given the test setup presented in 6.1.1, the following parameters are used during the evaluation in this project.

**Number of publishers** – This parameter is fixed for all tests conducted in this thesis project. Each test case is constructed with one single publisher which is launched at the C1 machine and act as the primary distributor of data during the experiments.

**Number of subscribers** – The second parameter concerns the number of subscribers in the system which request information provided by the publisher. By performing tests with varying number of subscribers, it is possible to get an indication of how well the implementation scale when the number of clients
requesting certain information increase. For the sake of proving the capabilities of the implementation all subscribers are registered for the same type of events.

**Amount of data** – To be able to measure system behaviour when the workload increase, the amount of data sent from the publisher is a parameter which vary between tests.

**Type of data** – Besides the messages transmitted over the network during registration to the server, the data communicated during the tests are events designed for evaluation purposes. These events are sent as packages with the size of 114 bytes.

6.2 **Performance tests**

Performance evaluations may include several types of measurements on different parts of the system. Due to the specified requirements about reduced time for message handling and load on sender side, tests are conducted with focus on latency and CPU load.

As the evaluation is conducted in a simulated environment where several subscribers run on the same host the timing may be affected, causing differences in the result. To provide a higher degree of reliability, each test is repeated 10 times to reduce the potential differences. From these iterations an average can be calculated along with the standard deviation.

6.2.1 **Latency**

Since low latency has been identified as a key aspect in time-critical systems, like the one examined in this thesis, the new implementation and the old version is evaluated in terms of latency. For this purpose, the delay between sending messages and receive responses is measured as well as the sending time for a publisher node.

**Time for publishing to N subscribers**

The first test aims to investigate how the sending time for a publisher is affected by the introduction of multicast in the system. For this test, the time it takes on the sender side to publish a set of events to N number of interested subscribers is measured.

Based on the examined system in this project, which contains up to 20 nodes, the experiments are conducted with subscribers in the range of 5-40 to cover a possible future scenario where the system increase in size. The same test is performed both using the new implementation and the original system, running unicast for message distribution, to be able to compare the different communication strategies. By measuring the time, it takes for one publisher to send 5000 messages to all subscribers, an average and standard deviation is calculated which is presented in Table 3. To give a clearer overview of the relationship between the two implementations the average publish time is plotted in a graph and the result can be seen in Figure 14.
Table 3: Average publishing time and standard deviation for multicast and unicast

<table>
<thead>
<tr>
<th># Subscribers</th>
<th>Average publish time (ms) multicast</th>
<th>Standard deviation (ms) multicast</th>
<th>Average publish time (ms) unicast</th>
<th>Standard deviation (ms) unicast</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>55.63</td>
<td>4.87</td>
<td>273.95</td>
<td>0.76</td>
</tr>
<tr>
<td>10</td>
<td>58.13</td>
<td>10.88</td>
<td>550.42</td>
<td>1.58</td>
</tr>
<tr>
<td>15</td>
<td>55.42</td>
<td>7.51</td>
<td>827.10</td>
<td>1.45</td>
</tr>
<tr>
<td>20</td>
<td>53.80</td>
<td>2.67</td>
<td>1103.48</td>
<td>1.92</td>
</tr>
<tr>
<td>25</td>
<td>52.84</td>
<td>0.23</td>
<td>1379.64</td>
<td>2.74</td>
</tr>
<tr>
<td>30</td>
<td>53.76</td>
<td>1.18</td>
<td>1657.40</td>
<td>5.84</td>
</tr>
<tr>
<td>35</td>
<td>53.72</td>
<td>1.40</td>
<td>1932.10</td>
<td>3.72</td>
</tr>
<tr>
<td>40</td>
<td>53.22</td>
<td>0.47</td>
<td>2208.10</td>
<td>1.57</td>
</tr>
</tbody>
</table>

Moving the replication process out on network elements implies that the sender only needs to publish one instance of an event, which have the potential of reducing the time for message handling on sender side. Publishing to a group address as used in the multicast implementation will thus lead to an almost constant sending time, independent of number of subscribers interested in the current topic. This is clearly reflected in the graph of Figure 14 where it also becomes evident that the original communication design based on unicast have a nearly linear increase in time as the number of subscribers is incremented.

Figure 14: Average publish time for the new and old implementation
**Round-trip time**

As the second latency test the round-trip time (RTT) is measured for one sender to publish an event and get a response from each subscriber. Similar to the experiment for measuring sending time, the number of subscribers is the variable in this experiment. The same range of subscribers (5-40) is used during the experiment and is incremented by five between test runs.

Since neither the new implementation nor the original system provides the functionality of low-level acknowledgements, the response message is an ordinary event which is sent directly to the node which is the source of the incoming message. More specifically, the publisher sends an event to all registered subscribers who, upon receiving the event, creates a new event and send this message directly to the publisher. The time measured in this test is the difference between first packet leaving the publisher until last response is received. For this purpose, tcpdump is used for capturing incoming and outgoing traffic. The RTT for one sender to publish to \( N \) subscribers is estimated according to the formula presented below.

\[
RTT = t_{N:th \ response} - t_{first \ published \ packet}
\]

The result of these measurements can be seen in Table 4 and Figure 15 which presents the average RTT for both the multicast and unicast implementation. Also, in Table 4 the standard deviation is presented in microseconds.

<table>
<thead>
<tr>
<th># Subscribers</th>
<th>Average RTT (µs) multicast</th>
<th>Standard deviation (µs) multicast</th>
<th>Average RTT (µs) unicast</th>
<th>Standard deviation (µs) unicast</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>551,8</td>
<td>62,8</td>
<td>621,0</td>
<td>74,0</td>
</tr>
<tr>
<td>10</td>
<td>676,1</td>
<td>128,6</td>
<td>711,2</td>
<td>130,8</td>
</tr>
<tr>
<td>15</td>
<td>728,8</td>
<td>132,6</td>
<td>781,7</td>
<td>96,5</td>
</tr>
<tr>
<td>20</td>
<td>821,2</td>
<td>125,9</td>
<td>880,0</td>
<td>157,5</td>
</tr>
<tr>
<td>25</td>
<td>912,9</td>
<td>148,4</td>
<td>1035,4</td>
<td>193,0</td>
</tr>
<tr>
<td>30</td>
<td>1019,2</td>
<td>217,7</td>
<td>1103,3</td>
<td>219,4</td>
</tr>
<tr>
<td>35</td>
<td>1091,9</td>
<td>210,0</td>
<td>1139,9</td>
<td>281,7</td>
</tr>
<tr>
<td>40</td>
<td>1193,3</td>
<td>232,3</td>
<td>1438,3</td>
<td>365,0</td>
</tr>
</tbody>
</table>
No clear difference in RTT can be seen from the presented result. However, the last measurement, with 40 subscribers, indicates that the difference between unicast and multicast will widen if the number of subscribers increase. The fact that the amount of messages sent during this test is quite low in comparison with the previous test and that relatively powerful desktop computers are used, makes the result not that surprisingly. To provide a better comparison between the two implementations the RTT could be measured for a larger set of messages. Since the unicast implementation have to handle almost twice as many events as the multicast solution, it will most certainly increase the difference between the two of them.

6.2.2 CPU usage
The last performance test aims to measure how the CPU load is affected by high message-intensity. This test is conducted by successively increase the number of messages/second and measure the percentage CPU usage for the machine running the publisher. For these measurements the Linux command *top* is used to capture the CPU usage during a time interval of five seconds. The time interval was chosen since a first iteration showed stabilized values for the CPU usage at that time. During this experiment 40 subscribers are used to receive messages from the publisher in order to make the workload for the sender as high as possible. In Figure 16 and Table 5 the result from the CPU tests is presented.
Table 5: Average CPU usage and standard deviation for multicast and unicast

<table>
<thead>
<tr>
<th>Message/second</th>
<th>Average CPU usage multicast (%)</th>
<th>Standard deviation (%)</th>
<th>Average CPU usage unicast (%)</th>
<th>Standard deviation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>3.74</td>
<td>1.61</td>
<td>16.96</td>
<td>2.51</td>
</tr>
<tr>
<td>5000</td>
<td>5.35</td>
<td>1.65</td>
<td>44.09</td>
<td>2.00</td>
</tr>
<tr>
<td>10000</td>
<td>6.06</td>
<td>1.53</td>
<td>44.10</td>
<td>2.20</td>
</tr>
</tbody>
</table>

By studying the result presented in Table 5 it becomes clear that the multicast implementation outperforms the unicast version of the system significantly. This indicates that the procedure of sending one instance of a message to each subscriber is costly for the unicast implementation. As the system is used in a context where large amount of data is handled this is an evident drawback of the unicast solution. Looking at the graph in Figure 16 the CPU usage for 10000 message/second is almost the same as 5000 messages/second for the unicast implementation which is unexpectedly. An explanation to this might, however, be that the iteration of subscribers increases the delay between messages which makes the CPU less loaded.

Figure 16: Average CPU usage for the new and old implementation
7 Conclusions

The thesis presents the development and evaluation of a multicast communication design which can provide group communication in a single transmission to reduce time and required resources for message handling. The project is about solving the problem situation which may arise due to increased data flow in distributed systems, and highlights the importance of choosing an appropriate communication strategy for the specific purpose and needs of a system.

While the gap between data transmission volumes and network capacity tend to widen, it can be noted that systems of embedded character composed of small devices with limited resources increase. In order for these systems to handle an increased data-intensity, adaption of the communication strategy is essential to meet the requirements of an application. Limited resources on endpoints in combination with insufficient use of the network is likely to decrease the performance and potential of a system, which has been established by studying a real-life example within industry.

In the case of point-to-multipoint communication, where identical information is consumed by several, traditional data transmission methods based on unicast may not be able to deliver quality of service. Redundant traffic on the outgoing link and an expensive internal process for distributing data to large groups of receivers, motivates the need of additional strategies in this context. Applying multicast as a substitute to unicast in systems based on a publish/subscribe architecture has proven to be an efficient solution for decreasing both time for message handling and workload on sender side.

The project has resulted in a communication design built upon multicast, which is implemented in a distributed electronic warfare system, for self-protection in military aircrafts. The prototype implementation provides continued support for all system functions and allows for data distribution to a set of interested subscribers in a single transmission with the use of multicast groups. This is achieved by extracting the replication of messages out on switches. In this way, information is delivered with less effort from the publisher and without intermediate forwarding, which is in-line with the purpose and the goal for the project. It could also be determined that the new design removes the need of the external registration to the server to make publishers aware of what subscribers are interested in specific topics. With the use of multicast groups which are mapped to specific domains and a MAC-address table on switches, publishers are no longer required to know which subscribers to delivery messages to.

From a scalability point of view, it can further be established that the new implementation prepares the system for increased data exchange by better utilizing the available resources, as identical data is no longer transmitted over the same link several times. This, together with the described method for
realizing the implementation reflects the main question of the thesis which could be considered as answered.

From the result of the evaluation it is determined that the multicast design outperforms the original system based on unicast in all the conducted tests. The most significant differences are observed in the publishing- and CPU-tests where the strategy implemented in this project clearly have the potential of increasing the performance of the system. With an almost constant time for distributing a set of messages, independent of the number of subscribers, makes multicast an attractive alternative to unicast in terms of scalability. Moreover, the CPU-test shows a tremendous difference between the two implementations. The process of iterating through all available subscribers and make a separate call to the socket handling object each time has proven to be a costly process. The fact that the multicast solution requires much less CPU-usage for performing the same task, compared to the unicast implementation, further motivates the use of multicast over unicast in this context.

7.1 Discussion

The development of the prototype has included an extensive process of defining properties and limitations of the current system to be able to realize the implementation. This has led to a communication strategy which gives the system several advantages over the original version based on unicast. The replication and distribution of messages is now handled by hardware on switches which is both faster more fault-tolerant than having the application on sender side dealing with this process. As previously mentioned this have the potential of decrease the workload on endpoints in the system as well as decrease the time for the delivery process.

Moreover, as the registration phase between clients and the server now becomes redundant, it could potentially reduce the start-up time for the system if it is removed. A potential drawback of the design is, however, that the creations of multicast groups is based on domains instead of events. This may lead to some redundancy as multicast messages could be sent to a node with an interest in a domain but with no services interested in the current event. Preferably, to get a more precise message distribution, multicast groups should be created on event-level. Handling of incoming messages is also limited to a single thread in the application which may cause latencies when the data distributor should forward messages to services.

The results of the evaluation were in-line with the expectations. Nevertheless, the CPU-tests could preferably be re-conducted with fine-tuned parameters to provide more accurate results. Also, the RTT-test could be re-conducted with increased number of subscribers and messages to show the potential of multicast in a more real-life scenario.

Considering the non-functional requirements which was defined for the prototype, a discussion about how they can be maintained with the new implementation is appropriate. Testability and observability could be achieved
by connecting to a port on a switch and collect all traffic currently circulating which allows for complete monitoring of the system. The aspect of timeliness is considered due to the use of UDP-based communication which reduces the latency since no low-level acknowledgement is sent between sender and receivers. Low latency is thus achieved at the cost of less reliability which should be kept in mind during the design of a communication strategy to ensure that the implementation still fulfils the underlying requirements of a system.

When it comes to the choice of methods for the project it could be established that the action research model has been an appropriate research strategy for the purpose of this degree project. Since the project was based on a qualitative approach with focus on integration of theory and practise this was a suitable method to be able to meet the goal of the project. Conducting a pre-study of the stakeholder’s system as the initial phase of the project was necessary in order to identify the current problem situation. It provided useful insights both for establishing requirement and for the implementation phase of the project.

Using an agile working process for providing flexibility in the development phase has proven to be useful when replacing an existing design with a new implementation. Continuous testing, to ensure that the system still can provide its original functionality has been an important aspect during implementation. A less strict model for software development has also allowed for changed requirement as the project progressed. The experiments were set up in order to evaluate and confirm tentative hypothesis derived from the initial phase of the project by comparing the new implementation with the original communication strategy. In accordance with the purpose and goal for the project, tests were conducted with focus on time for message handling and workload. Having the opportunity to compare the implementation with an existing design was valuable to demonstrate the achievements of the project.

### 7.2 Future work

This project has focused on the peer-to-peer communication between publishers and subscribers, which only constitutes one part of all messages being sent in the system. For example, heartbeat messages sent by the server is a continuously ongoing process which is not considered in the implementation of this project. Heartbeat messages are sent repeatedly to all nodes in the system in order to monitor available resources, and is a process which is currently using unicast for message delivery. Proposed future work is therefore to implement multicast even for the heartbeat functionality in the system. This extension would require similar logic to be implemented on the server, including an appropriate definition for multicast groups which nodes can subscribe for. Potentially, this could increase the overall performance of the system as the server becomes less loaded.

Future work could also include an evaluation in the test-rig provided by the stakeholder, to investigate the potential of multicast in a more real-life scenario. The experiments presented in this thesis are conducted in a laboratory environment, based on relatively powerful desktop computers. Therefore, it
would be of interest to investigate how the two implementations perform compare to each other in an embedded environment with less powerful units.

Furthermore, as the implementation is based on the UDP transport protocol it would be interesting to evaluate the system from a reliability perspective. By measuring e.g. percentage of packet loss between publisher and subscribers it could indicate whether there is a need for some kind of recovery mechanism for retransmission of messages and how this would affect the communication network.
References


