FFI-RAPPORT

18/00904

Traffic control in a heterogeneous mobile tactical network with autonomous platforms

Lars Landmark
Erlend Larsen
Øivind Kure

Traffic control in a heterogeneous mobile tactical network with autonomous platforms

Lars Landmark
Erlend Larsen
Øivind Kure

Norwegian Defence Research Establishment (FFI) 15 August 2018

Keywords
Eksperimentering
Kommunikasjonsteknologi
Kommunikasjonsprotokoller
Kommunikasjonsnettverk

FFI-rapport
18/00904

Project number
1367

ISBN
P:978-82-464-3020-1
E: 978-82-464-3021-8

Approvers
Espen Skjelland, *Research Director*
Jan Erik Voldhaug, *Research Manager*
Bjørnar Libæk, *Acting Research Manager*

The document is electronically approved and therefore has no handwritten signature.

Summary

Future military operations will involve autonomous platforms and systems. In order for these systems to fully achieve their potential, the autonomous platforms need to collaborate. Each platform might have its own independent role in the autonomous system, but they will all work together to reach the same common goal. Hence, an efficient military autonomous system will be highly dependent on a communication network where traffic control is required to better utilize the resources. Through traffic control, the network can discriminate between several levels of importance, routing traffic along different performing network paths.

Our motivation for this work was to gain experience with Software Defined Networking (SDN) as a tool for designing and experimenting with new network functionality in an environment where the radios are developed for operational use. Our objective was to gain experience by running SDN along with traditional routing in a heterogeneous network consisting of autonomous platforms. SDN was used as a tool to employ traffic control, so that we could intercept and further traffic engineer-specific traffic along with ordinary routing. Different traffic types were forwarded dependent on the capabilities of the radio networks. Our testbed consisted of different autonomous platforms, i.e. Unmanned Aerial Vehicle (UAV), Unmanned Ground Vehicle (UGV), and Unattended Ground Sensor (UGS). An experiment with the testbed was performed at Rygge Aerodrome where most elements of the testbed were tested, except physical elevation of the UAV platform. Through the experiment, several challenges were identified.

In collaboration with Kongsberg Defence & Aerospace (KDA), we have implemented and tested SDN as a tool for traffic control using UAV. SDN with the help of the OpenFlow protocol and the Ryu SDN controller was installed by KDA on one of their broadband military radio models. Based on this implementation we could write our own network functionality, and thus easily customize traffic control. OpenFlow provides us with a tool where we can customize network functionality at a low cost in terms of coding hours. In our experiment, the traffic was relayed by intermediate nodes, due to the distance and traffic type, either on the ground or in the air. Military Ultra High Frequency (300–3000 MHz) (UHF) and Very High Frequency (30–300 MHz) (VHF) frequency bands were used for connectivity. OpenFlow was used for traffic control within the UHF radio network, and between the UHF and the VHF radio networks. By using OpenFlow we were able to customize how traffic should be handled by the network.

This report is mainly written for network designers interested in SDN and autonomous systems in the tactical domain. The report focuses on design choices in an SDN network and on our observations in an experimental SDN network.

Sammendrag

Fremtidige militære operasjoner vil innbefatte bruk av autonome plattformer og systemer. For at disse systemene skal kunne utnyttes fullt ut, må de autonome plattformene samarbeide. Hver plattform kan ha sin egen uavhengige rolle i det autonome systemet, men for å nå et felles mål må de arbeide sammen. Et effektivt militært autonomt system vil derfor være svært avhengig av en kommunikasjonsinfrastruktur hvor trafikkontroll vil være avgjørende for å utnytte ressursene. Gjennom trafikkontroll kan nettverket skille mellom ulike viktighetsnivåer og rute ulik type trafikk over nettverksstier med ulik ytelse.

Vår motivasjon med dette arbeidet var å få erfaring med Software Defined Networking (SDN) som et verktøy for å designe og eksperimentere med ny nettverksfunksjonalitet i et miljø der radioene er utviklet for operativ bruk. Vårt mål var å få erfaring ved å kjøre SDN sammen med tradisjonell ruting i et nettverk bestående av autonome plattformer. SDN ble brukt som et verktøy for trafikkontroll. Ved hjelp av SDN kunne vi fange opp og trafikkstyre data av interesse samtidig som vanlig ruting håndterte resten av trafikken. Ulike trafikktyper ble videresendt avhengig av radionettverkets muligheter. Vår testbase besto av forskjellige autonome plattformer: ubemannede luftfarkoster (UAV), ubemannede landkjøretøy (UGV) og bakkesensor (UGS). Et eksperiment med testbasen ble gjennomført på Rygge flystasjon hvor de fleste elementene i testbasen ble testet, bortsett fra fysisk elevering av UAV-plattformen. Gjennom eksperimentet ble det avdekket en rekke utfordringer.

I samarbeid med Kongsberg Defence & Aerospace (KDA) har vi implementert og testet SDN som et verktøy for trafikkontroll ved bruk av UAV. SDN ble implementert av KDA ved hjelp av protokollen OpenFlow og Ryu SDN-kontroller. Basert på denne implementasjonen kunne vi skrive vår egen nettverksfunksjonalitet, og dermed lett tilpasse trafikkontrollen. OpenFlow gir oss et verktøy hvor vi kan tilpasse nettverksfunksjonaliteten innenfor relativt få kodetimer. I vårt eksperiment ble trafikken videresendt av mellomliggende noder, på grunn av avstand og trafikk, enten på bakken eller i luften. Radiobåndene for militær Ultra High Frequency (300–3000 MHz) (UHF) og Very High Frequency (30–300 MHz) (VHF) ble brukt for sammenkobling av enheter. OpenFlow ble brukt til trafikkontroll innenfor UHF-radionettverket, og mellom UHF- og VHF-radionettene. Ved å bruke OpenFlow kunne vi tilpasse hvordan trafikk skal håndteres av nettverket. Dette ble demonstrert ved at spesifikk trafikk ble omdirigert over UAV.

Denne rapporten er hovedsakelig skrevet for nettverksdesignere som er interessert i SDN og autonome systemer i taktisk domene. Rapporten fokuserer på designvalg og våre observasjoner i et eksperimentelt SDN-nettverk.

Contents

Preface

The experiment described in this report was part of the final experiment for the Norwegian participants in the multi-national research project Coalition Network for Secure Information Sharing (CoNSIS) phase II Task 1. It was a combined effort between the FFI project "Ubemannede kjøretøy for Forsvaret", Kongsberg Defence & Aerospace (KDA) through their projects on CoNSIS and Robotics, and the authors, both as part of the FFI project "Taktisk mobil kommunikasjon for Forsvaret" and as part of the FFI project "Autonomi for ubemmanede systemer".

1 Introduction

An efficient military autonomous system in the future will be highly dependent on a communication infrastructure facilitating traffic control. With traffic control, we mean the ability to control on which links to utilize for selected data flow(s) at a chosen time. A communication infrastructure is needed to interconnect the autonomous entities. The communication infrastructure may consist of multiple radio networks operating on different frequencies bands and hence, functionality to control where to forward specific traffic is highly required. Without traffic control, we will not be able to utilize the available resources. However, functionality for traffic control is not well supported in existing tactical radio equipment. To advance the research and potential implementation of traffic control in military radio equipment, tools and concepts such as Software Defined Networking (SDN) should be investigated. SDN is a concept associated with the benefits of a decoupled control plane and forwarding plane. A network is built up of routers and/or switches interconnected with network cables or network radios. Traffic between any two end points require control logic to instruct how the traffic should be forwarded. The control logic is located in the control plane, while the actual data is forwarded within the forwarding plane. In a traditional architecture, the control and forwarding plane are located in each router/switch, while in SDN they are decoupled. That is, the control plane is moved out from the switches and further placed in an external controller that enables to control multiple switches. The benefit of the controller is its programmability. Hence, network functionality can be programmed in the controller.

A benefit of implementing traffic control through SDN and OpenFlow is the flexibility of adding new functionality in software, given that the software is open. The software is open, i.e., the end user can write its own policy rules and further apply them to the network, which is often the case for switches and routers supporting OpenFlow. In our experiment, we used SDN as a tool for traffic control. Our goal was not to control all data traffic by SDN, but only to redirect a few flows. The underlying motivation was to use ordinary routing as the primary control engine for traffic control, but SDN as a tool for redirecting traffic where applicable. Our goal was therefore to redirect traffic over the UAV if ordinary routing had decided to forward over UGV. In other words, we used routing functionality implemented by the radio vendor and SDN to traffic engineer specific traffic. As a consequence, if the SDN controller failed due to an error, and stopped working, the traffic rules implemented by the controller would stop working, and the standard routing would take over, although without the desired traffic engineering functions.

Testbeds have long been a popular method for testing new concepts. However, implementing new concepts onto operational military equipment has historically been difficult, even for small additive changes. Traffic forwarding through a heterogeneous network is often based on static or dynamic routing, either with or without routing metrics. In our setting a heterogeneous network is a network consisting of two radio networks, a UHF and VHF radio network. Traffic control function could be implemented through traditional router configuration using link weights or through other functions. However, it could be simpler to implement completely new network functionality, compared to adapting and forcing traditional alternatives to a new concept. For the experiments described in this report, we therefore chose to investigate SDN as a new way to implement network policies. The OpenFlow [1] protocol, an SDN-based standard, provides us with an open Application Programming Interface (API) towards the network nodes. It is well-suited for policy services. This way, we are not dependent on getting access to internal radio network software.

Radio network communication requires an interdisciplinary approach between application writers and network designers. In our case, we used an Unmanned Aerial Vehicle (UAV) as a network relay between two ground platforms. The UAV is a sparse network resource due to its limited energy reserves. Hence, there is a need to know how this resource can be utilized without detailed knowledge of the network topology. Therefore, as a starting point, we planned an experiment in the field to showcase possible functions, discover pitfalls and learn valuable lessons. Our contribution in this experiment was the implementation of network functionality to support network usage policies.

In this report, we describe our network topology and design choices and our experience with our testbed. The approach chosen, i.e. SDN used for traffic control, was very easy to work with and easy to understand in our small testbed. The lessons learned through the approach can hence be of value to others considering using SDN for tactical communication.

This report is organized as follows. In Chapter 2, the concept of SDN is introduced, presenting generic challenges and design choices for using SDN in mobile tactical networks. In Chapter 3, the communication building blocks of the testbed are described, and in Chapter 4, an SDN design discussion for our experiment is presented. In Chapter 5, we provide our observations, before a conclusion and future work ends the report in Chapter 6.

2 Software Defined Networking

SDN is normally associated with the benefits of decoupling the forwarding plane and the control plane, an open API and the introduction of a central coordinator. Out of these three, the main benefit is a decoupled control plane and data plane, contrary to traditional networking. The control plane is where the routing protocol is running, and the forwarding plane is where the actual traffic is forwarded within a network node. In traditional Internet Protocol (IP)-networking (Figure 2.1) each router or switch runs their own local control software (control plane), which further dictates how the forwarding plane should forward traffic. Each node in the network exchanges control information with other nodes. This information is further used to build a forwarding table. Hence, each node calculates its own forwarding plane as a distributed process, according to an algorithm set by the routing protocol.

In an SDN network (Figure 2.2), the control plane may be moved out from the network node and further moved into a centralized node called a controller. The controller may control more than one network node/switch at the same time. Hence, the control plane of the network switches can be centralized in one node that can control multiple switches at the same time. Each switch keeps the ability to receive instructions on how to forward traffic. The flow rules on how to forward traffic are handled through flow tables and group tables within each switch. A flow rule consists of *Match* conditions and *Actions*, which provides a forwarding policy for a particular flow. *Match* identifies traffic to be handled by this rule, while *Action* describes how to handle the traffic. The traffic can be identified with different granularity, from a more coarse granularity identifying packets by netmask, to a more fine granularity where a specific packet is identified e.g., by IP source address and the packet's unique identification field.

SDN and especially OpenFlow [1] has gained popularity as a tool to enable network automation and policy enforcement, especially within data centers. It provides an architecture with which each operator can write customized policy rules depending on time, participants and network environments. Such functionality would be beneficial also in wireless mobile networks. However, wireless networks do not have the same characteristics as their wired network counterpart. Wireless mobile networks often experience topology changes, and links are prone to error. The end equipment is often designed to be energy conservative and is hence implemented with low Central Processing Unit (CPU) power. Consequently, there is a tradeoff as to where and how many controllers to be configured in a tactical network. The decision on the number of controllers and their location depends on the network size, bandwidth availability, CPU power, controller availability, and the requirements for network reliability. This report is more focused on how we can experiment with traffic control using SDN in a military heterogeneous network, and hence program network functionality.

Tactical military networks will likely see more SDN equipment deployed in the future. The main reasons are the open API, and decoupled control and forwarding plane. These two features provide the network owner the ability to design and implement traffic control without access to vendor proprietary software. Also, in near future we are likely to see solutions for the support of both traditional routing and SDN technologies within the same routing domain. At present, there are few solutions for the exchange of state or route information between SDN and legacy routing. One problem is maintaining consistent forwarding rules as a consequence of forwarding decisions taken at the IP level (routing) and at the switch level (SDN).

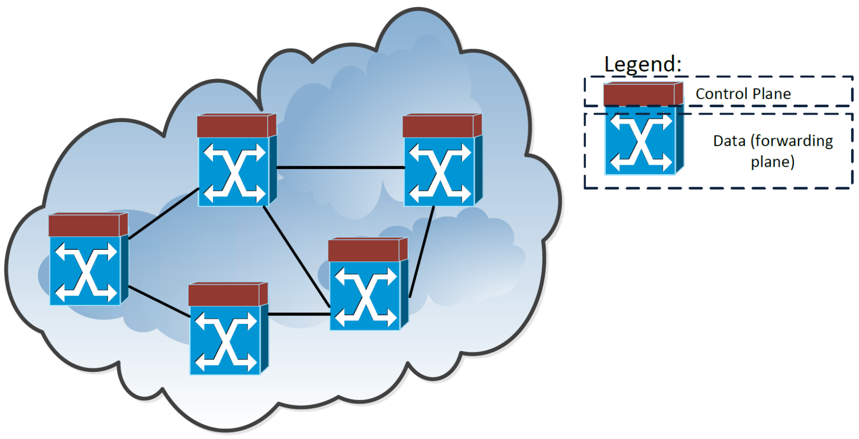


Figure 2.1 Traditional networking architecture, from [2].

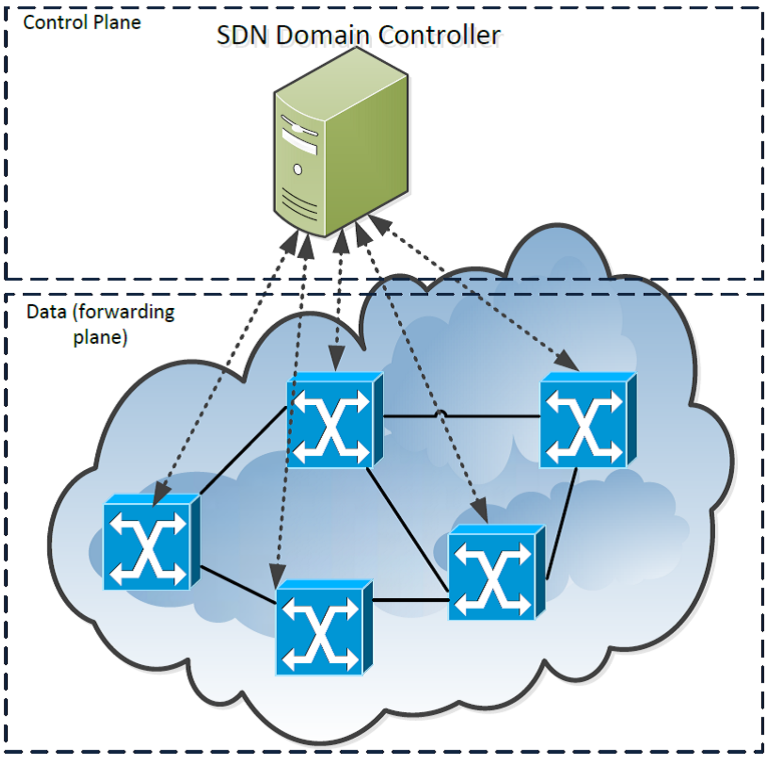


Figure 2.2 SDN architecture, from [2].

2.1 SDN and Workflow

Evaluating new ideas typically involves three phases: the design phase, the implementation phase, and the testing/evaluating phase. Shifting from traditional Linux software routing to OpenFlow does not provide large benefits in the design phase, but more so in the implementation phase. First, we could easily experiment with military radio equipment, without necessarily having access to the development environment (e.g., the radio software and its code library), as this is often vendor closed. Also, the time spent on implementing functionality is reduced as a consequence of using a platform independent code library, since this could easily be moved between different platforms.

Controller functionality can be more easily reused and tested within different development environments. The evaluation of ideas is often performed either by simulation, emulation, testbed, or in a combination of these. Each development platform has its own libraries and might require code customized for the specific development environment. SDN and OpenFlow, on the other hand, provide us with the ability to develop functionality without the need of tailored code for the specific environments. Hence, by migrating to SDN, we can develop code more independent of the environment and more easily test the same code within all domains (simulation, emulation, and testbed).

3 Testbed building blocks

Through the experiment, we tested a flexible communication concept between a deployed ground sensor and a Ground Control Station (GCS). The test included a Unmanned Ground Vehicles (UGVs) which also had relay capability. In the future, and in our design, UGVs will likely have the ability to deploy sensors. The deployed sensors, e.g., the Unattended Ground Sensor (UGS), accumulated sensor data that could be transmitted directly or relayed via an intermittent network node back to a ground station. Data could be carried over different radio equipment depending on the distance, capacity requirements and/or other policy rules. We employed two types of platforms, one ground-based and one elevated. In our experiment, there were three equally equipped ground platforms, and an additional elevated platform (referred to as an Unmanned Aerial Vehicle (UAV)). In our case, we carried data over both Ultra High Frequency (300-3000 MHz) (UHF) and Very High Frequency (30-300 MHz) (VHF) on all ground-based platforms. The UAV was only fitted with a UHF radio.

The UAV was added to improve the radio coverage, and thus the network stability of the UHF multi-hop network. Improved stability is achieved due to less link change because of longer transmission range, and the ability to position the UAV according to communication needs. Improved capacity is achieved because of improved link stability due to Line-of-Sight (LoS), and thus less packet loss due to low link quality. In a generic network, the UAV would have a larger interference range, and thus could reduce the network throughput due to reduced spatial reuse.

Our testbed communication infrastructure (Figure 3.1) consisted of two different types of mesh radios, one narrowband radio (VHF), and one broadband radio (UHF). The main differences between the two radios were the longer range and the lower data-rate for the narrowband radio. The three ground nodes were equipped with both radios, while the UAV was only equipped with the broadband radio, due to weight constraints. Our experiment was based on a radio family close to the radios used by the Norwegian Army for the VHF band. The Multi-Role Radio (MRR) radio system is used within the Norwegian Armed Forces (NAF) for communication. The manufacturer of the radio, Kongsberg Defence & Aerospace (KDA), produces two versions of this radio, one for the NAF, and one for the international market. In our experiment, we used an international version of the MRR radio. The main difference between the national and international version is the security implementation. We also used a radio not procured by the Norwegian Army produced by KDA operating in the UHF band named WM600, UM600, and SR600, commonly referred to as the KDA TacLAN radio suite.

KDA installed a standard Open vSwitch (OVS) bridge in their UHF radio within this project. The OVS is a software switch with SDN support. The OVS bridge transfers the Medium Access Control (MAC) layer packets to and from the IP routing process. The OVS implementation comes with the support of OpenFlow version 1.3. Thus, it was a simple operation to retrofit OpenFlow into an existing military radio platform. The code from the experiment is available in an FFI-internal document [3].

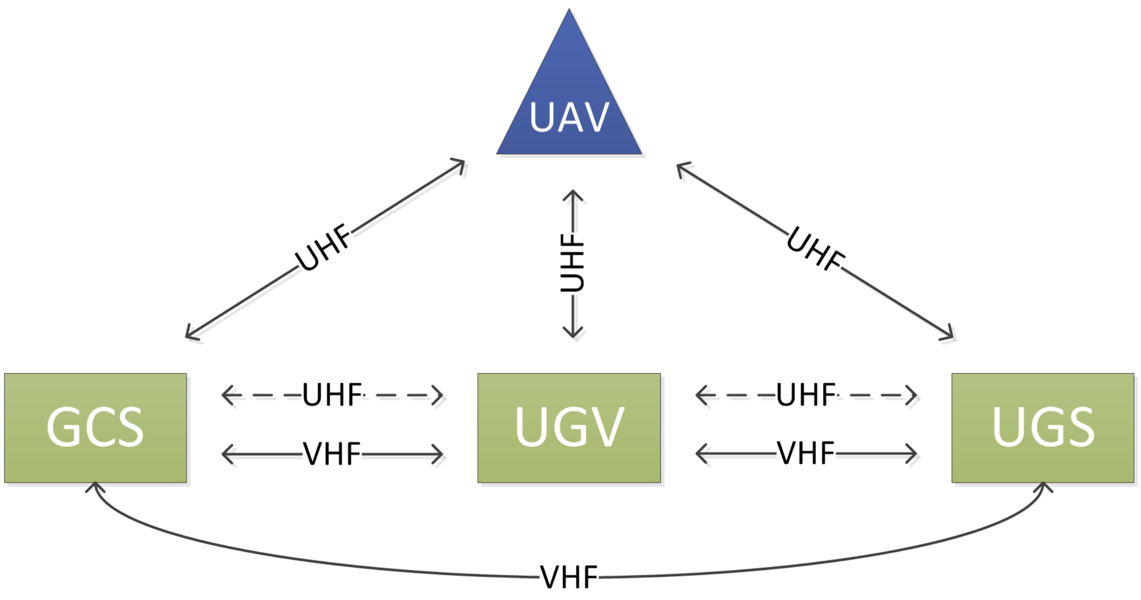


Figure 3.1 Experiment components and communications. Dashed lines indicate presumed unreliable links.

4 Design choices for our communication network

The overall goal of the experiment was to use SDN for traffic control and run it alongside traditional routing. The combination of SDN and traditional routing give us flexebility to customize traffic control for all or some traffic if required. SDN was used as a tool to employ traffic control rules, so that we could intercept and control specific traffic flows while ordinary routing was running. Different traffic types were forwarded dependent on the capabilities of the radio networks. Hence, customized policy rules were written per radio. In this chapter, we discuss our SDN design choices. Our network equipment consisted of a combination of SDN-enabled hardware and legacy hardware not supporting SDN with limited computational capability. Because of our hardware combination and traffic engineering goal, we needed to address four problems, presented in the following subchapters:

1. SDN controller placement
2. Combining traditional routing and SDN
3. Topology discovery
4. SDN flow rules

4.1 SDN controller placement

In the literature, the design of SDN is often illustrated with one controller, controlling multiple switches at the same time. However, a large network can usually not be operating with one controller alone, due to a number of reasons, e.g.: the traffic load on the controller, the delay between the network switch and the controller, and robustness in case of controller failure. A central controller in mobile wireless radio networks will most likely not scale, and thus is considered less applicable. There are a number of problems associated with the concept of a central controller in wireless mobile radio networks. The first problem is the availability of a central controller, due to mobility, partitioning and, last but not least, error prone links. Transmission failure is more common within normal operation in wireless environments than in wired networks. A central controller receives and sends all signaling traffic. This design require certain control on the delay from switch and controller and back, and the volume of the signaling traffic. In the wired domain, SDN is typically deployed with the SDN control traffic being Out-of-Band, using separate Ethernet ports and links (i.e., a separate network) to connect forwarding devices to the controller and exchange control traffic. In this setting, SDN control traffic will not compete with data traffic, and hence the delay and the traffic volume is defined. In the wireless domain, data and signaling traffic would typically operate In-Band due to the lack of a dedicated radio system. The result is that the control traffic delay is less predictable since it is influenced by additional factor such as data traffic and the wireless link quality compared to the wired domain. Furthermore, a central controller sees all control traffic. That is, the controller is either source or destination for all control traffic. Hence, the control traffic will add to the problem of congested channel as approaching the central controller. Concentrating all control traffic on one wireless channel makes SDN with a central controller less scalable and predictable in the wireless domain.

Although a centralized controller sees one topology, the switches will move and have temporarily misconnections to the controller. The environment is changing rapidly in mainly three dimensions: topology, offered traffic and available resources to forward traffic. In order for a controller to calculate accurate and consistent forwarding decisions, the controller would need accurate topology information consistent with the other controllers. Furthermore, the controller also needs to ensure that the validity time is within the projected validity time of the present topology information. Hence, a plausible implementation is to implement network rules with a hard expiration time according to the expected network change. The timer can be implemented in the controller or in the switch. In the case of controller implementation, the benefit is reduced control traffic. The drawback is that the controller will run the chance of not reaching the switch, either in the case of lost control packets (a logically partitioned network/path error) or in the case of a physical partitioned network. A time limited rule in the switch will allow for the possibility of searching for an available alternative controller in case of lost connectivity to a controller, at the cost of a higher amount of signaling traffic.

We chose to run one controller at each ground platform (GCS/UGV/UGS) due to the problem of state exchange between the controllers and the computational capacity/CPU clock frequency at the UHF radio. Initially, we ran a controller on each UHF radio on each ground platform, but it was seen early that the radio could not offer the computation resources required to process rule changes. Hence, the Operating System (OS) caused a backlog which again caused packet loss. Our SDN switch (OVS-switch [4]) on the UHF radios was configured to send packets without a matching rule up to the controller. In case a burst of packets arrived to the switch, all packets were sent up to the controller and the controller had to process each of them individually, which is CPU intensive. Our SDN controller did not block or buffer successive packets while processing the first packet, including installing rules.

Our controllers were running isolated, and installed flow rules independent of the state of the other controllers. A consequence of running multiple controllers is that the controllers might have different topology views, for instance due to a lost topology packet. This problem is not specific for SDN running multiple controllers, but general to distributed routing protocols. In an architecture using a central controller, we would have one topology view and calculate paths accordingly. In case of a lost topology packet, the consequence would be that the links would not be used.

4.2 Combining traditional routing and SDN. Where to locate the switch within a node?

IP routing was the main forwarding engine in our experiment, and thus SDN was only used for traffic engineering where the SDN policy did not match the IP forwarding. Our goal was therefore to send traffic up to the routing layer, and further intercept and eventually change IP forwarding decisions. In our test, the protocol Wireless Open Shortest Path First (WOSPF) [5] was used for routing. Hence, we needed to install our software switch after IP routing. Figure 4.1 shows the forwarding path within the UHF radio nodes supporting both routing and SDN. The interface *eth0* was located on the wired network, while uhf0 was our UHF radio interface. The OVS bridge (software switch) was installed between *uhf0* and IP/Linux routing. In our case, our OVS switch was installed with two port entries. One connected to the UHF radio interface *uhf0* and the second port to the IP routing. The default operation was to forward packets between the two ports. Hence, if no SDN rules were installed, the OVS switch did not change the traffic forwarding. In order to use SDN to redirect traffic, traffic rules were to be installed dictating traffic to be sent up to the controller. By this design, we were able to write policy usage rules for traffic within the UHF network.

Within the UHF radio network the traffic was classified into two classes. The first class was forwarded by traditional routing WOSPF, while the second class was intercepted and forwarded by SDN in case the SDN forwarding and the traditional routing did not match. The intercepted

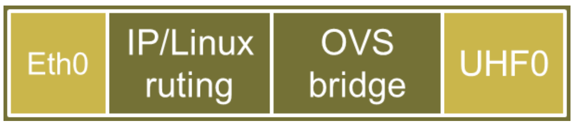


Figure 4.1 The packet forwarding path within the UHF radios.

traffic forwarded by SDN was calculated by shortest path between the controllers. In our case, the static policy was set to prefer the UAV, as it was assumed more reliable, at the cost of increased interference at the ground. A consequence of this policy was an added likelihood of increased path length in those cases where traditional routing suggested a shorter path, resulting in higher resource consumption.

4.3 Topology discovery

In our test, we did not have any SDN controller controlling more than two switches, but we had more than one controller. In the SDN architecture, the Link Layer Discovery Protocol (LLDP) and the Broadcast Domain Discovery Protocol (BDDP) are the two topology discovery protocols often cited within the SDN literature. In SDN, the switches are known by the controller, but the links among the switches are not initially known. In general, the LLDP and BDDP both send link discovery messages on each port announcing their switch ID. The receiving switches report back to the controller on which port/interface the message was received. Based on this information, the controller can build up a topology view that may, for instance, be used for calculating shortest path.

In our experiment, we ran multiple controllers and required topology discovery among the controllers, but not among the switches. Consequently, neither the LLDP nor the BDDP were applicable for our testbed. Instead, we found it more time efficient to implement our own topology discovery method within SDN. The cost was higher bandwidth consumption over UHF, compared to gaining access to the WOSPF topology database. Our topology messages were only sent over UHF radio, due to limited bandwidth over the narrowband radio system (VHF). We only redirected traffic within the UHF radio network. Thus, it was not required to send control messages over VHF radio. As a consequence, our testbed ran two topology control modules: WOSPF and our custom written topology discovery daemon. WOSPF was running in the background and its topology database ought to have been used for route calculation for both WOSPF and SDN. Software to acquire and read the Quagga WOSPF topology database has later been written in a master assignment [6]. One of the author's observations was that it would require extensive knowledge of the deployed routing protocol to gain access to read topology information.

4.4 SDN flow rules

In our testbed, we had one deployed sensor delivering a wide-angle image, zoom images, and telemetry. The different traffic flows had different network requirements and were thus marked with different IP Differentiated Services Code Point (DSCP), (DSCP value 10, 20 and 30).

Due to the problem of mobility and stale links, we decided to implement the SDN rules with a hard timeout, at the cost of higher load on the controller. As earlier mentioned, we were aware of the limited CPU power of the UHF radio, but nevertheless initially implemented an onboard controller controlling only the onboard UHF radio switch. It resulted in many packets sent up to the controller before the controller was able to set flow rules. This resulted in approximately 20 packets being sent up to the controller at each rule timeout, and thus clearly showed throughput variance. As a solution to the problem, we had three design alternatives:

1. Extend the timeout at the cost of added time to detect stale links.
2. Implement a stateful link state. In case the topology discovery detected a link change, the rules associated with the stale path/link were changed. This solution was tested (in lab) and resulted in more stable throughput at the cost of added code/state complexity.
3. Keep our initial hard timeout and introduce an external controller with sufficient CPU power directly connected on the platform Local Area Network (LAN) (Figure 4.3). This became our design choice, due to a more simple and clean design, compared to the previous stateful design.

Our design for redirecting the traffic was based on rewriting the packet's MAC destination address. In cases where the traditional routing within the UHF radio (Figure 4.1) found the UGV as the next hop, and our SDN control logic found a path through the UAV, the computer Ryu-controller installed a rule to forward traffic up to the UAV by rewriting the packet MAC destination address on UHF radio OVS-switch. As a result, we were not dependent on Address Resolution Protocol (ARP) support in our SDN code. In traditional routing, IP packets are redirected by next hop IP address and hence sent down to ARP. ARP is then responsible for assigning the MAC address to the IP packet associated with the next hop IP address. In our testbed, we knew the next hop MAC address (acquired by the topology discovery daemon), and could rewrite the packet's MAC addresses without involving ARP. Operating in environments involving legacy switches and SDN switches requires rewriting both the source and the destination MAC addresses. Otherwise, there is a risk of altering the state in the legacy switch, resulting in incorrect switching of data packets.

OpenFlow prevents traffic from being sent out on the arriving port. Our computer, where the Ryu-controller was installed, consisted of one Ethernet-interface and one installed OVS-switch. The goal was to rewrite the MAC addresses and thus send the packet out on the same port as it arrived. In our case it meant that packets arriving on the Ethernet-interface were sent up to the connecting port on the OVS-switch. The OVS-switch then rewrote the MAC-destination address before sending the packet out on the same arriving port down to the Ethernet-interface. OpenFlow does not allow packets being sent out on the arriving port ([4] Section B.6.3 IN PORT Virtual Port page 105), but it supports virtual ports on the same physical interface, allowing the same packets to be sent over the virtual interface associated with the incoming physical interface.

Our design was prone to routing loops caused by SDN and the native routing protocol pointing at each other. The UHF and VHF radio networks were connected by our SDN traffic control module. Both the UHF and the VHF radio networks ran routing protocols, exchanging information about reachable subnets and their routing cost. Thus, when the VHF routing protocol was made aware that a destination was reachable through the UHF network, it would bounce the packets to this subnet back towards local UHF radio, thereby overriding the SDN forwarding decision. Our solution to this was to turn off routing on the Ethernet interfaces of both radio types (the wired network side) and perform SDN switching within the wired domain.

4.5 Resulting communications testbed

This section describes our resulting communication testbed, including the communications network, the platforms and the internal connectivity on each platform. A more detailed description on the code design, and the code itself, is available in an FFI internal document [3].

Looking closer at the communication infrastructure in our experiment (Figure 4.2), each of the ground nodes included a computer running an SDN controller and an OpenFlow-capable switch. A simple learning switch, denoted X, interconnected the controller, the terminals and the (UHF/VHF) radios. Furthermore, each of the UHF-radios ran an OpenFlow-capable switch. The elevated node operated only over UHF.

Figure 4.3 shows our communication configuration for the stationary ground platforms. Due to practical issues, the switch denoted X in the center of the figure was a standard learning switch, while the two SDN-controlled OVS-switches were placed on the UHF-radio and in the computer.

The *Application computer* was a computer performing application tasks outside of the network, such as collecting, processing and exchanging sensor information. The *Application computer* was configured with a default route entry to the Ryu-controller located on the *Computer*.

The *Computer* ran the Ryu-controller software [7]. The *Computer* was operating as the platform's SDN controller and controlled two OVS switches: one on board the *Computer* and one onboard the UHF-radio. The *Computer*'s OVS-switch was responsible for switching traffic between the UHF and VHF networks, and hence forwarded traffic to the UHF or the VHF radio. This instruction was installed into the OVS-switch on the *Computer* by the Ryu-controller. The UHF switch was used to traffic engineer traffic over the UHF radio network.

The application computer was responsible for collecting sensor information. The sensed information was further marked depending on the information, and forwarded towards the *computer* as the next IP hop. The traffic was marked by changing the DSCP value in the IP header and used for traffic control. Three DSCP values were used; DSCP value of 10 was used for VHF traffic, 20 and 30 was used for UHF traffic, while 30 was used for UAV.

The central traditional switch denoted X was selected because of practical issues. We had two options: SDN or a legacy switch. A legacy switch was chosen instead of an SDN switch, due to our SDN-switch's out-of-band control channel architecture and the single Ethernet port available on the controller. (The *Computer* which ran the controller also needed to communicate with the switch on the UHF-radio, not only communicate with the central switch.) In the SDN architecture,

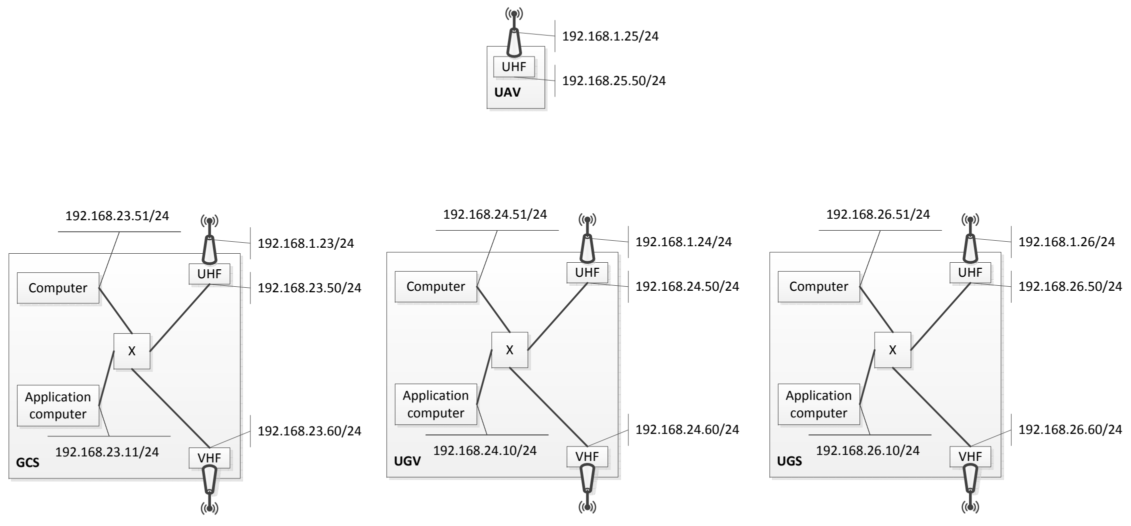


Figure 4.2 The network architecture.

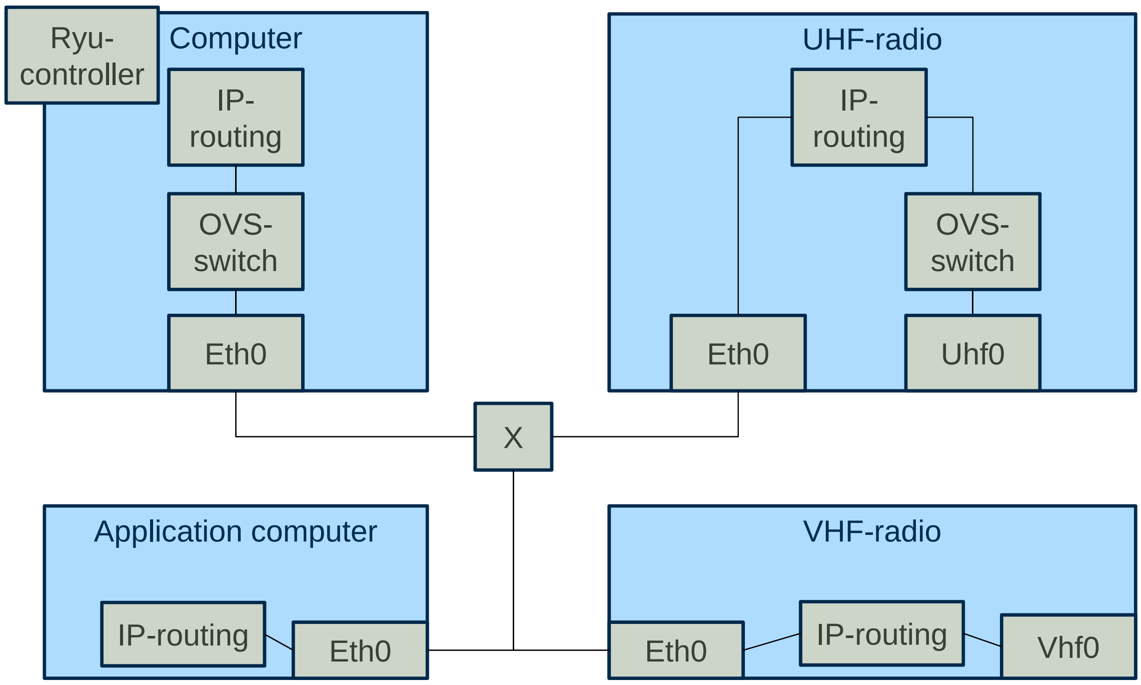


Figure 4.3 The ground node communication architecture.

out-of-band is normally used for the control traffic between the switch and the controller when using traditional routing. A typical design is to assign one or more ports/interfaces with traditional control layer software. That is, one or more ports on a switch keeps a traditional control layer implementation such as spanning tree enabling the switch to connect to the controller after bootup.

If we were to use the SDN switch, we had to either have two interfaces on the computer, which we did not have, or arrange it with Virtual Local Area Network (VLAN). Our solution was instead to install an OVS-switch and the Ryu-controller on the computer and route all traffic towards this node.

5 Experimenting with OpenFlow as a testbed platform

Moving from designing and implementing solutions to experimenting with the resulting code was in itself a learning experience. Several of the design choices described in the previous chapter are results of the testbed work in a laboratory environment with a radio network topology emulator. The first tests were focused on confirming the topology discovery functionality. Next, we evaluated the correctness of 1) setting traffic rules in each switch, 2) the traffic load generated by these rules, and 3) the communication between the switch and the controller. At this stage, the traffic was generated using mgen scripts [8]. After establishing the proper behaviour by the controllers and the switches, traffic from the actual applications was introduced. Recorded application data was available, so the system could be tested with data played back in real time. The final experimentation was performed at Rygge Aerodrome, where the applications and platforms were deployed outdoors, producing real data.

The results presented in this chapter focus on the final experiment, where the design choices had been established, and most – but not all – bugs had been sorted out. In summary, the implemented code worked as expected, except for one bug which triggered erroneous traffic rerouting in the legacy switches. Two issues that were unrelated to the SDN software did also present some challenges. They are elaborated on below.

Using Wireshark [9], we were able to confirm that the traffic generated with a DSCP value of 10 was forwarded with help of SDN over the MRR radio network (VHF). The traffic types were identified by the logs, and showed that packets that were received on either the GCS or the UGS platforms with a DSCP value of 10 were forwarded via the MRR radio network. Packets with a DSCP value different from 10 were forwarded via the UHF radio network (WM-600 radio).

We were also able to confirm that traffic with a DSCP of 30 was routed over the UAV according to the design. However, we were not able to confirm the behavior of routing packets marked with DSCP of 30 over the UAV, while the other traffic was routed over the UGV. In our network design, we expected two hops within the UHF radio network: one path over the UGV and another over the UAV. While this topology occurred during our experiment, it was not stable, due to dynamic routing. Some control messages were transmitted further than assumed, and as a result established a time varying topology switching between 1-hop and 2-hop between the UGS and the GCS. Consequently, instead of having two 2-hop paths (one over the UAV and one over the UGV) the 1-hop connection was often used.

The time varying topology switching between 1-hop and 2-hop presented us with an unexpected topology, and also an unwelcome link flapping. To some degree, this was a product of the lack of an elevated node (elevated high enough). This led to elaborate "tuning" of the radio parameters, where the expected relays (the UAV and the UGV) had to be configured with a more robust link rate than the end nodes (the UGS and the GCS). The most important and problematic link was that of the direct link between the UGS and the GCS. This impacted the routing protocol on the UHF-radios, resulting in periodic link loss and rerouting with a considerable timeout (~40 s). Link flapping and/or link stretching is a familiar observation in radio networks. Routing protocols built on shortest path will prefer a 1-hop path over a 2-hop path. Normally, it is better to route traffic over a less reliable 1-hop link than over a 2-hop path. However, in situations where a 2-hop path is reliable, but a 1-hop links is going up and down, shortest path routing will often switch back to the 1-hop path if possible. In such situations, a link might be inserted into the forwarding table, but cannot be used for data forwarding. As a result, traffic will not reach the destination before the link times out, and the 2-hop path is selected. During the lifetime of the 1-hop link, the capacity of the path between source and destination is thus reduced. This problem was not addressed by us in advance, but we could have mitigated the problem by our traffic engineering mechanisms. As manual operator we trusted our UAV over both 1-hop (direct link), and 2-hop via UGV, and could therefore redirect eligible traffic over UAV. This problem is often mitigated by solutions such as the Expected Number of Transmissions (ETX) routing metric. Combining a routing metric and traffic engineering involving humans could be preferable in our situation.

When analyzing the log files after the conclusion of the experiment, some unexpected behavior was discovered: the controller computer at the UGS had received numerous packets destined to the UGS application computer. The computers at the UGS were interconnected using a simple MAC-switch. However, the SDN code that was implemented for forwarding traffic from the sensor application computer to the MRR radio did not change the MAC source address for packets relayed from the controller computer to the MRR radio. The result was that every time the controller computer at the UGS forwarded a packet to the MRR radio, the traditional switch updated its MAC address mapping to point to the controller computer for the sensor application computer MAC address. Traditional switches are self-learning, and store which ports to reach which packet sources. We did not rewrite the MAC source, but only the destination and hence, the switch did point back to the last node it received this MAC address which was our controller node and the SDN switch. As a consequence, the computer became a black hole for traffic coming from the radios to the sensor application computer. Packets that were not SDN switched but received from the same source would correct the switch mapping, since these packets were routed normally by the controller computer included setting a new MAC source address.

Our Kongsberg TacLAN (WM/UM/SR600) radios were fitted with a software version optimized for a single-sender scenario. It was made to work better with video transfer, and thus the MAC retransmissions functionality was switched off. Furthermore, the radio was configured so that it discarded the packet ready for transmission if the medium was found to be "busy". This posed a problem, both in a relay situation and in a bidirectional transmission situation, and the configuration was disadvantageous in our setting. Our application computer was delivering large segments of data resulting in fragmentation. In a relay situation and bursty traffic due to large packets fragmented to fit the Maximum Transmission Unit (MTU), the second fragment from the source would likely coincide in time with the relay's transmission of the first fragment. With the current radio MAC behavior, one of the fragments would be discarded, leaving the destination lacking this fragment. Without an end-to-end transmission protocol that takes responsibility for retransmission, the entire data segment would be lost. On recommendation from KDA, a fix was implemented which delayed the transmissions from the expected relay nodes by 100 ms±50 ms. In the lab, this implementation gave some wanted effect, in that the packet loss was reduced. In essence, this fix reduced the collision probability and thereby reduced the packet loss at the cost of throughput. The cost of throughput was that the maximum transmission rate was reduced. The problem of discarding packets if the medium was found "busy" was not handled and if lost packet was detected based on missed acknowledgment, nothing was done.

As a consequence of radio problems, our planned performance tests were discarded. As an end result we were able to control traffic while combining traditional routing and tailored traffic forwarding using SDN and OpenFlow in a heterogeneous network. Traffic was initiated by the application node and sent by a default route towards the computer running SDN. Traffic was then routed and sent back with new destination mac address depending on being traffic for UHF or VHF network. Traffic sent to the UHF network and filtered for preferred UAV was also verified to be sent over the UAV.

6 Conclusion

With the emergence of autonomous platforms and sensors in the Norwegian Armed Forces, traffic engineering should be employed to more effectively exploit the already limited network resources. In the experiment traffic engineering was sought done using Software Defined Networking (SDN) instead of by traditional routing metric control. OpenFlow and the Ryu-controller was installed onto Kongsberg military radios in collaboration with KDA, so that we could write our own software for traffic control. As a result, we implemented traffic control software on a generic platform (the Ryu-controller), that was further used on all the ground platforms, as well as on the UAV.

In this experiment we gained experience with running SDN along with traditional routing in a small tactical network consisting of autonomous platforms. It worked as expected and is a viable solution for further studies operating in a larger network. Ordinary military radios were running traditional traffic forwarding, but were in addition configured with SDN used for intercepting and rerouting traffic in case a local decision found it more suitable. The benefit of this design is the failover mechanism, i.e., if the SDN controller stopped working, the standard routing protocol would still be functioning, although without the desired traffic engineering functions.

Our experience with SDN and wireless mobile networks is that it is less suitable compared to SDN within the wired domain. The main reasons are availabe link data rate and CPU capacity and path reliability. The combination of our OpenFlow 1.3 implementation, low CPU capacity, low link capacity and path reliability made us to reduce the distance, in number of hops, between the switches and the SDN controller.

Through our work, we have seen that implementing new network functionality is relatively easy and may reduce the implementation time. Thus SDN may also be a powerful tool when working with traditional locked-in radios where altering the network behavior depends on the vendor's cooperation. By using SDN, we were able to access the data flow plane in order to control the traffic, without requiring direct access to the radio software. Nor did we require recompilation or reinstallation of the software for each incremental test. Hence, SDN was beneficial in terms of testing new functionality within ordinary military radio equipment.

Abbreviations

API	Application Programming Interface
ARP	Address Resolution Protocol
BDDP	Broadcast Domain Discovery Protocol
CoNSIS	Coalition Network for Secure Information Sharing
CPU	Central Processing Unit
DSCP	Differentiated Services Code Point
ETX	Expected Number of Transmissions
GCS	Ground Control Station
IP	Internet Protocol
KDA	Kongsberg Defence & Aerospace
LAN	Local Area Network
LLDP	Link Layer Discovery Protocol
LoS	Line-of-Sight
MAC	Medium Access Control
MRR	Multi-Role Radio
MTU	Maximum Transmission Unit
NAF	Norwegian Armed Forces
OS	Operating System
OVS	Open vSwitch
SDN	Software Defined Networking
UAV	Unmanned Aerial Vehicle
UGS	Unattended Ground Sensor
UGV	Unmanned Ground Vehicle
UHF	Ultra High Frequency (300-3000 MHz)
VHF	Very High Frequency (30-300 MHz)
VLAN	Virtual Local Area Network
WOSPF	Wireless Open Shortest Path First

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[2] E. Sørensen, "SDN used for policy enforcement in a federated military network," Master's thesis, Norwegian University of Science and Technology, Department of Telematics, 2014. [Online]. Available: https://brage.bibsys.no/xmlui/handle/11250/263070

[3] L. Landmark, E. Larsen, and Ø. Kure, "Traffic control in a heterogeneous tactical network using SDN: code listings," Norwegian Defence Research Establishment (FFI), FFI-notat 17/16390, 2017.

[4] Open Networking Foundation, "openflow-switch-v1.3.1," 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf

[5] K. Holter, A. Hafslund, F. Y. Li, and K. Øvsthus, "Design and implementation of wireless OSPF for mobile ad hoc networks," in *Scandinavian Workshop on Wireless Ad-hoc Networks (ADHOC 06)*, 2005.

[6] H. M. Fagervoll, "SDN in Heterogeneous Mobile Tactical Networks," Master's thesis, Norwegian University of Science and Technology (NTNU), 2017. [Online]. Available: https://daim.idi.ntnu.no/masteroppgave?id=16953

[7] Ryu. Accessed 04-May-2017. [Online]. Available: http://osrg.github.io/ryu/

[8] Multi-Generator (MGEN). Last accessed: 2017-05-10. [Online]. Available: http://www.nrl.navy.mil/itd/ncs/products/mgen

[9] Wireshark. Accessed 2017-12-06. [Online]. Available: https://www.wireshark.org/

About FFI

The Norwegian Defence Research Establishment (FFI) was founded 11th of April 1946. It is organised as an administrative agency subordinate to the Ministry of Defence.

FFI's MISSION

FFI is the prime institution responsible for defence related research in Norway. Its principal mission is to carry out research and development to meet the requirements of the Armed Forces. FFI has the role of chief adviser to the political and military leadership. In particular, the institute shall focus on aspects of the development in science and technology that can influence our security policy or defence planning.

FFI's VISION

FFI turns knowledge and ideas into an efficient defence.

FFI's CHARACTERISTICS

Creative, daring, broad-minded and responsible.

Om FFI

Forsvarets forskningsinstitutt ble etablert 11. april 1946. Instituttet er organisert som et forvaltningsorgan med særskilte fullmakter underlagt Forsvarsdepartementet.

FFIs FORMÅL

Forsvarets forskningsinstitutt er Forsvarets sentrale forskningsinstitusjon og har som formål å drive forskning og utvikling for Forsvarets behov. Videre er FFI rådgiver overfor Forsvarets strategiske ledelse. Spesielt skal instituttet følge opp trekk ved vitenskapelig og militærteknisk utvikling som kan påvirke forutsetningene for sikkerhetspolitikken eller forsvarsplanleggingen.

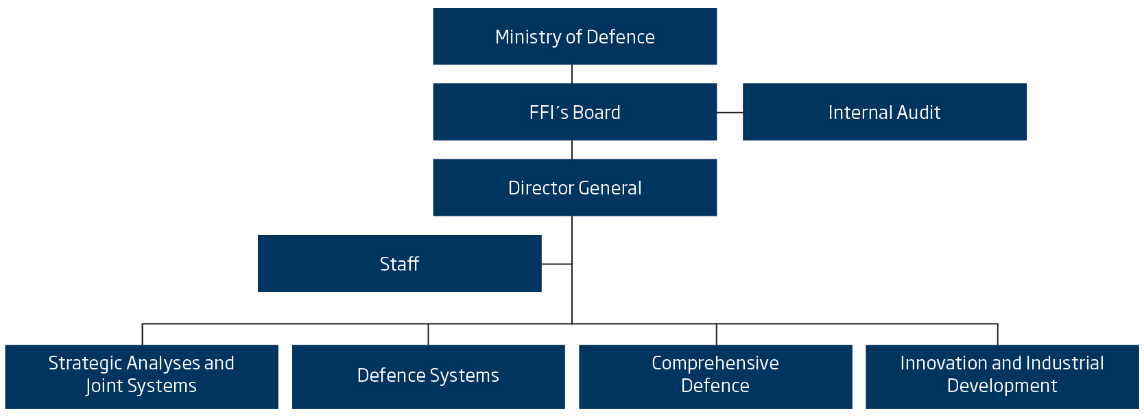
FFIs VISJON

FFI gjør kunnskap og ideer til et effektivt forsvar.

FFIs VERDIER

Skapende, drivende, vidsynt og ansvarlig.

FFI's organisation


Ministry of Defence
FFI's Board
Internal Audit
Director General
Staff
Strategic Analyses and Joint Systems
Defence Systems
Comprehensive Defence
Innovation and Industrial Development