UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

SDN for Forsvarets taktiske kommunikasjonsnoder

Masteroppgave

Geir-Roar Bakken

30.04.2015

«SDN kan gi økt operativ effekt for Forsvarets kommunikasjons infrastruktur (FKI)»
- cyberdomene - bildet om sammspill +god forståelse
+ drift av nettverk
? status-quo? svakheter i dagens nett?
- IP-teknologi for kommunikasjonslaget
+ god forklaring av grunnleggende
- kontroll og styring av trafikk, interoperabilitet, økt mobilitet
? tallfestes? fokus: enabling, monitoring, …
- overbruk av tiden i presentasjonen
+ tjenestekavlitet for militær prioritet
? data strøm for kvalitettsikring

SDN for Forsvarets taktiske kommunikasjonsnoder

Geir-Roar Bakken

geirroab@student.matnat.uio.no

30 april 2015

? attributes for policies (network, datarater, tjenestesuksess,…)
? vesentlig bedre styring

2015

SDN for Forsvarets taktiske kommunikasjonsnoder

Geir-Roar Bakken

http://www.duo.uio.no/

Trykk: Reprosentralen, Universitetet i Oslo

Sammendrag

Forsvaret skal gjennom et materiellprosjekt ta frem en kommunikasjonsløsning, Taktisk kommunikasjonsnode, som har behov for et bredt spekter av kommunikasjonstjenester, blant annet en løsning for styring og overvåking.

Software Defined Networking (SDN) er et konsept innenfor nettverksdesign hvor kontroll- og data planet er separert og logikken i nettverket blir sentralisert i en eller flere programvarebaserte tjenere kalt kontrollere.

I denne masteroppgaven har jeg gjennomført en studie som tar for seg hvordan SDN kan brukes for autentisering og aksesskontroll for Forsvarets taktiske kommunikasjonsnoder. I tillegg, hvordan policy for differensiert tjenestekvalitet kan bli implementert for tjenester som har behov for militær prioritet i Forsvarets kommunikasjonsinfrastruktur.

Ved hjelp av litteraturstudie i tillegg til design, implementasjon og validering av en løsning i et testoppsett med bruk av Cisco sin SDN arkitektur onePK, har jeg vist at SDN kan gi ny funksjonalitet for nettverks management i Forsvarets kommunikasjonsinfrastruktur.

Abstract

The Norwegian Armed Forces has ongoing projects which will take forward a communication solution, Tactical Communications Node, which will require range of communications, including network management services.

Software Defined Networking (SDN) is a concept within network design where the control- and data plane is separated and the logic of the network is centralized in one or more software developed servers called controllers.

In this thesis, I have conducted a study which examines how SDN can be used for authentication and access control for tactical communications nodes.
In addition, how policy for differentiated service quality can be implemented for services that need military precedence in The Norwegian Armed Forces communication infrastructure.

Using literature study in addition to design, implementation and validation in a test bed using Cisco's SDN architecture onePK, I have shown that SDN may provide new functionality for network management services in The Norwegian Armed Forces communication infrastructure.

Forord

Denne oppgaven er skrevet som en del av mitt masterprogram innen Kommunikasjonssystemer ved Høgskolen i Bergen, avdeling for ingeniør- og økonomifag, institutt for elektrofag, og Universitetet i Oslo/Universitetssenteret på Kjeller fakultet for matematikk og naturvitenskap, institutt for informatikk. Oppgaven er skrevet ved Høgskolen i Bergen.

Jeg vil gjerne takke min veileder Knut Øvsthus for å ha tatt seg tid til å gi gode tilbakemeldinger under arbeidet mitt med oppgaven. Jeg har også satt veldig stor pris på våre uformelle samtaler i fagområdet. Jeg vil også takke veileder Josef Noll ved Universitetssenteret på Kjeller.

Spesiell takk til Trond Hermansen og Tore Kvamsøe ved Sjøoperativsambandsskvadron, som la til rette for at jeg kunne fullføre min utdannelse.
Jeg vil også takke min venn Jarle Husebø for hans støtte og hans rolle som motivator når enden virket veldig fjern. Selv med store tidsforskjeller og lange arbeidsdager klarer han alltid å avsette litt tid.

Tilslutt vil jeg gjerne dedikere denne oppgaven til mine foreldre og mine besteforeldre. Takk for gode råd gjennom min oppvekst og kontinuerlig støtte under mitt militære profesjonsstudium og masterstudium.

Innholdsfortegnelse

Figurer

AF 1: FKI Kjernenettverk
AF 2: Produksjonstjenester for kommunikasjonstjenester

BF 1: IntServ
BF 2: DiffServ
BF 3: Policykontroll
BF 4: Policykontroll og RSVP. Kommunikasjon mellom PEP og PDP ved hjelp av COPS

CF 1: ATM-nettverk
CF 2: MPLS Domenet
CF 3: MPLS og RSVP
CF 4: Signalering, DiffServ over MPLS

DF 1: XML script - nettverkstopologi

FF 1: Sertifikat autentisering - oppsett av **TLS**
FF 2: Konsollvindu fra RADIUS
FF 3: Konstanter på nettverkselementet

GF 1: Et utdrag fra Wireshark og ping

Tabeller

1. Introduksjon

«Despite the ancient roots and growing profusion of theories about communication, there is not a field of study that can be identified as communication theory».

Robert T. Craig, 1999

Dette kapitlet gir en oversikt over motivasjonen og problemstillingen for denne oppgaven. De påfølgende to kapitlene presenterer relevant teori.

1.1 Motivasjon

1.1.1 Revurdering av tradisjonell nettverksinfrastruktur

En av de største trendene innenfor kommunikasjon med den største umiddelbare effekten er skifte til mobil kommunikasjon [1]. Globalt så utgjør den mobile trafikken 21 prosent av all internett trafikk og den er rask økende [2]. Mobilt internett bruk er ventet å overgå tradisjonell nettbruk fra det som også er kalt «desktop» klienter i 2014/2015 [3]. Mobil kommunikasjon og nye trender har tett sammenheng med utviklingen i fagområdet kommunikasjonssystemer som innbefatter sentrale områder som nettverksteknologi, digital kommunikasjon, informasjonssikkerhet og datasikkerhet. Utviklingen her går hurtig og den tar stadig nye former. Nye behov blir fremtrengende når nye tjenester og kapasiteter blir tilgjengelig. To eksempler på dette er den økende bruken av skytjenester i norske virksomheter og tradisjonell tale over radio samband og linjebaserte kommunikasjonstjenester, som er blitt Internet Protocol (IP) -baserte tale, video og datatjenester. Skal det derfor legges en strategi for samhandlingen som det ønskes å implementere så er det flere ulike elementer å ta hensyn til. Elementer som mobilitet, sikkerhet, skalerbarhet og tilgjengelighet kommer ofte opp som viktige faktorer og påvirker valget av kommunikasjonsløsninger. De nye trendene og utfordringene som har oppstått er blitt en drivkraft for å revurdere tradisjonelle nettverksinfrastrukturer. Gjennom forskning og utvikling har et nytt nettverksparadigme, programmerbare nettverk, fått økt fokus i industrien.

1.1.2 SDN

En av de største innovasjonene innenfor programmerbare nettverksdesign *Software Defined Networking* (SDN), har de siste årene fått mye oppmerksomhet i nettverksindustrien. SDN er et konsept hvor kontroll og data planet er separert fra hverandre på nettverksenheter som rutere og svitsjer. Kontroll logikken i nettverket blir plassert i SDN kontrollere som kommuniserer med nettverksenhetene ved hjelp av egne kommunikasjonsprotokoller.
Jeg skal i denne oppgaven se på om noen av egenskapene til SDN kan benyttes for å understøtte behovene Forsvaret har i sitt prosjekt innenfor mobile og deployerbare kommunikasjonsløsninger, Taktiske kommunikasjonsnoder (TKN). Forsvaret må på lik linje med industrien videreutvikle kommunikasjonsløsningene sine, men må også ta hensyn til militære operasjonelle behov og det som utvikles i sivil- og militær industri.

1.2 Problemstilling

Det er gjennomført en del forskning og noen implementasjoner på SDN på ulike nivåer men SDN er foreløpig ikke særlig utbredt i det kommersielle markedet. Google er kanskje den største aktøren på markedet i dag med sin implementasjon av SDN nettverket «B4», en kombinasjon av den åpne kildekoden Quagga sammen med OpenFlow (OF) [4].
Det er mange temaer og mye forskning som gjenstår omkring temaet SDN og det er mange utfordringer som gjenstår i Forsvaret sitt prosjekt for Taktiske kommunikasjonsløsninger. *SDN for Forsvarets taktiske kommunikasjonsnoder* er derfor en sammensatt og langt fra en triviell oppgave. Denne oppgaven ser nærmere på hvordan SDN kan kontrollere aksess av noder som ønsker å benytte kommunikasjonstjenester i Forsvarets kommunikasjonsinfrastruktur (FKI). Oppgaven ser også på iverksettelse av QoS policy for TKN ved hjelp av SDN.

I hovedlinjer:

- Hvordan kan SDN benyttes til å håndtere tilgangskontroll for tjenester og brukere i nettverket?

- Hvordan kan SDN iverksette forhåndsdefinerte policyer for differensiert tjenestekvalitet for datatrafikk som har militær prioritet?

1.3 Avgrensing

På grunn av omfanget av oppgaven har jeg vært nødt til å gjøre noen avgrensinger. I problemstillingen ref. Kapittel 1.2: «*(…)å håndtere tilgangskontroll for tjenester og brukere (...)»,* ser jeg spesifikk på mekanismer i SDN for å kunne håndtere funksjonalitet for tilgangskontroll. Jeg drøfter ikke sikkerhetsarkitekturer og robustheten av mekanismene i min implementasjon.

1.4 Oppgavens struktur

Ny teknologi skal kunne gi ny funksjonalitet for å understøtte militære- operasjoner og konsepter og Forsvaret sine overordnende policyer og strategier. Jeg har derfor valgt å starte med en gjennomgang av noen utfordringer i nettverksarkitekturen som er relevante for Forsvaret, deretter litt om Forsvaret sin arkitektur og prosjektet Taktisk områdesamband. Videre blir SDN konseptet («ny» nettverksteknologi) og forskning gjennomgått før jeg drøfter skisserte problemstillinger som omhandler ny funksjonalitet med bruk av SDN. I drøftingen presenterer jeg forslag på SDN design som blir grunnlaget for min implementasjon av løsning som jeg evaluerer, tester og validerer. Jeg oppsummerer tilslutt i en konklusjon.

For å unngå at oppgaven ble for mange sider inneholder den kun noen utdrag fra JAVA-kode, scripts og konfigurasjonsfiler som ble utviklet for løsningen. Komplette koder og konfigurasjonsfiler kan gis ut på forespørsel til forfatteren av denne avhandlingen.

Oppgaven er organisert på følgende måte:

Kapittel 2 gir først bakgrunnsinformasjon om noen utfordringer i konvensjonelle nettverksarkitekturer som har relevans for pågående IKT prosjekter i Forsvaret. Deretter gir kapittelet bakgrunnsinformasjon om Forsvarets kommunikasjonsinfrastruktur, prosjektet Taktiske kommunikasjonsløsninger, og noen teknologier og tjenester som Forsvaret benytter og har relevans for denne oppgaven.

Kapittel 3 introduserer SDN konseptet.

Kapittel 4 gir en introduksjon til relevant forskning og utvikling og et multinasjonalt prosjekt.

Kapittel 5 drøfter bruken av SDN i FKI og presenterer problemstillingene i aktuelle scenarioer med mulige SDN design for Taktiske kommunikasjonsnoder.

Kapittel 6 beskriver forskjellige evalueringsmetoder for SDN og hvilken metode som ble valgt for denne oppgaven.

Kapittel 7 tar først for seg aktuelle SDN kontrollere og valg av kontroller teknologi. Implementasjonen og testoppsett blir deretter beskrevet.

Kapittel 8 beskriver testene, valideringen og resultatene.

Kapittel 9 gir en konklusjon på oppgaven og forslag til videre arbeid.

Forkortelser og akronymer er listet på side 116.

2. Bakgrunn

Jeg introduserer i dette kapittelet noen utfordringer i tradisjonelle nettverksinfrastrukturer som er relevant for Forsvaret, deretter litt om Forsvarets kommunikasjonsinfrastruktur og prosjektet Taktisk kommunikasjonsløsninger. Kapittelet gir også en kort beskrivelse av autoriserings-, autorisasjons- og regnskapstjenester, tjenestekvalitet og relevante nettverksteknologier. SDN er beskrevet i kapittel 3.

2.1 Behovet for en ny nettverksarkitektur

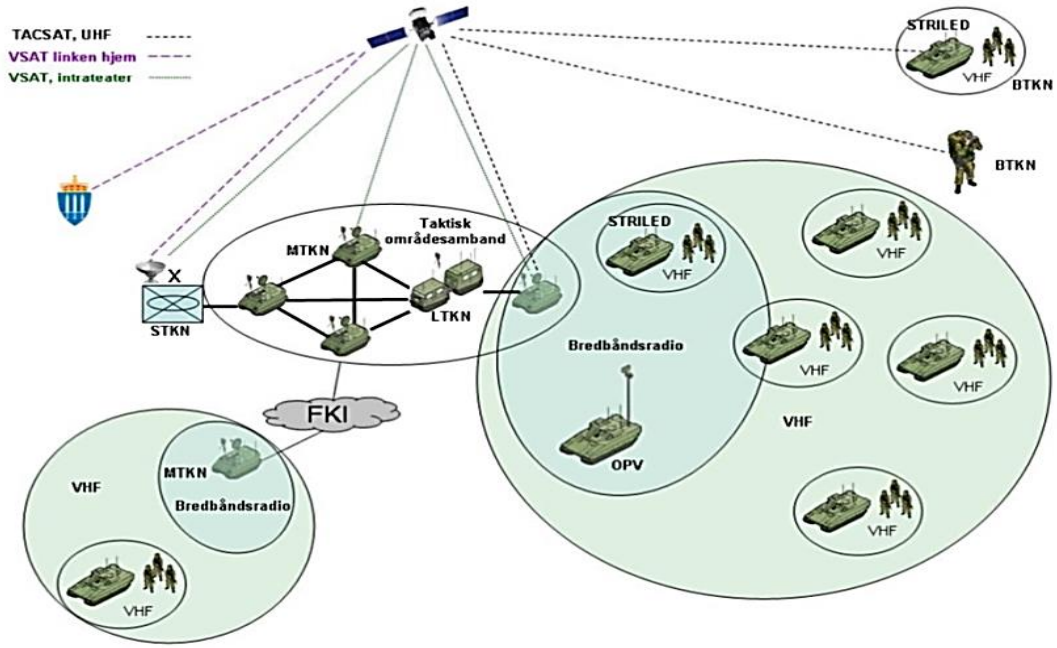
Mange konvensjonelle nettverk er hierarkisk oppbygget med nivåer av Ethernet-svitsjer ordnet i en trestruktur. Designet var fornuftig når klient-server databehandling var dominerende og nodene i nettverket var statiske, de endret ikke posisjon. Statisk arkitektur er ikke tilrettelagt nettverk hvor nodene beveger seg inn og ut av nettverket hele tiden, nettverkstopologien må derfor kunne tilpasse seg endringene som skjer.
Den statiske nettverksarkitekturen står i sterk kontrast til den dynamiske i dagens servermiljø, der virtualisering av servere har betydelig økt antall verter som krever nettverkstilkobling og fundamentalt endret forutsetninger om den fysiske plasseringen av vertene. Før virtualisering var applikasjonene residerte på en enkelt server og primært utvekslet trafikk med utvalgte brukere. I dag er programmer fordelt over flere virtuelle maskiner (VM) som utveksler trafikk med hverandre og migrerer for å optimalisere og rebalansere serverbelastninger. VM migrasjon utfordrer mange aspekter av tradisjonelle nettverk, fra adresseplaner, navnerom til den grunnleggende tanken om et segmentert ruting basert design.

Implementering av IP-basert teknologi for våre tale-, data- og videotjenester pågår i stor skala hos internettleverandører og tele- og mobilselskaper. «*100 prosent av landets husstander skal de nærmeste årene få tilgang til et bredbåndstilbud av grunnleggende god kvalitet(...)»[5].* Og i 2016 er teleoperatørene klar for IP-telefoni over 4G eller LTE [6]. Mens eksisterende IP-nettverk kan gi differensiert tjenestekvalitet nivåer for ulike applikasjoner, er klargjøring av disse ressursene ofte veldig manuelle. Vi må konfigurere hvert av leverandørens utstyr separat og justere parametere som nettverksbåndbredde og QoS på en per sesjon, per applikasjon basis. På grunn av sin statiske natur kan nettverket ikke dynamisk tilpasse seg endringene som skjer i trafikkmønstrene, programmene og brukerkravene. En transparent QoS policy for å oppnå ende til ende tjenestekvalitet er derfor meget vanskelig å få til.

Med implementering av IP for video- og taletjenester kommer også økningen av datatrafikk som i de siste årene har hatt en eksplosjonsartet vekst. Behovet for nettkapasitet spesielt til skytjenester, overføring av video og produksjon av tjenester som telefoni og nettløsninger, er eksempler på aktiviteter som krever spesiell behandling for å kunne overføres problemfritt. Tilgjengelig nettkapasitet når brukeren ønsker eller har behov for det er blitt et fokusområde i industrien og har de siste årene også blitt et fokusområde i Forsvaret.

2.1.1 Forsvaret – kapasitet og samhandlingsprosjekter

Forsvaret har flere pågående prosjekter for å kunne dekke det økende behovet for nettkapasitet [7] og for å optimalisere FKI (ref. Kapittel 2.2). De siste årene har Forsvaret innført en rekke nye stasjonære, mobile og deployerbare kapasiteter, for eksempel systemplattformer, sensorer og IKT-systemer. I tillegg gjennomføres det prosjekter for å forbedre samhandlingstjenestene som telefoni og videokonferanser. Forsvaret skal også gjennom prosjektet Taktiske kommunikasjonsløsninger, ta frem helt nye kommunikasjonsløsninger som skal erstatte det som tradisjonelt blir kalt feltsamband: Telefoni, radio, data og andre sambandsmidler i felt som benyttes mellom sjefer og deres avdelinger.
Feltsamband, eller taktisk områdesamband (TOS) består av mobile og bærbare kommunikasjonsnoder som sammen utgjør kjernen i et taktisk områdedekkende sambandssystem (figur 1). Systemene benyttes til øvelser og operasjoner i Norge og i utlandet, i tillegg til utdanning ved Forsvaret sine sambandsskoler.



Figur 1: Taktiske områdesamband[8].

2.2 Forsvarets kommunikasjonsinfrastruktur

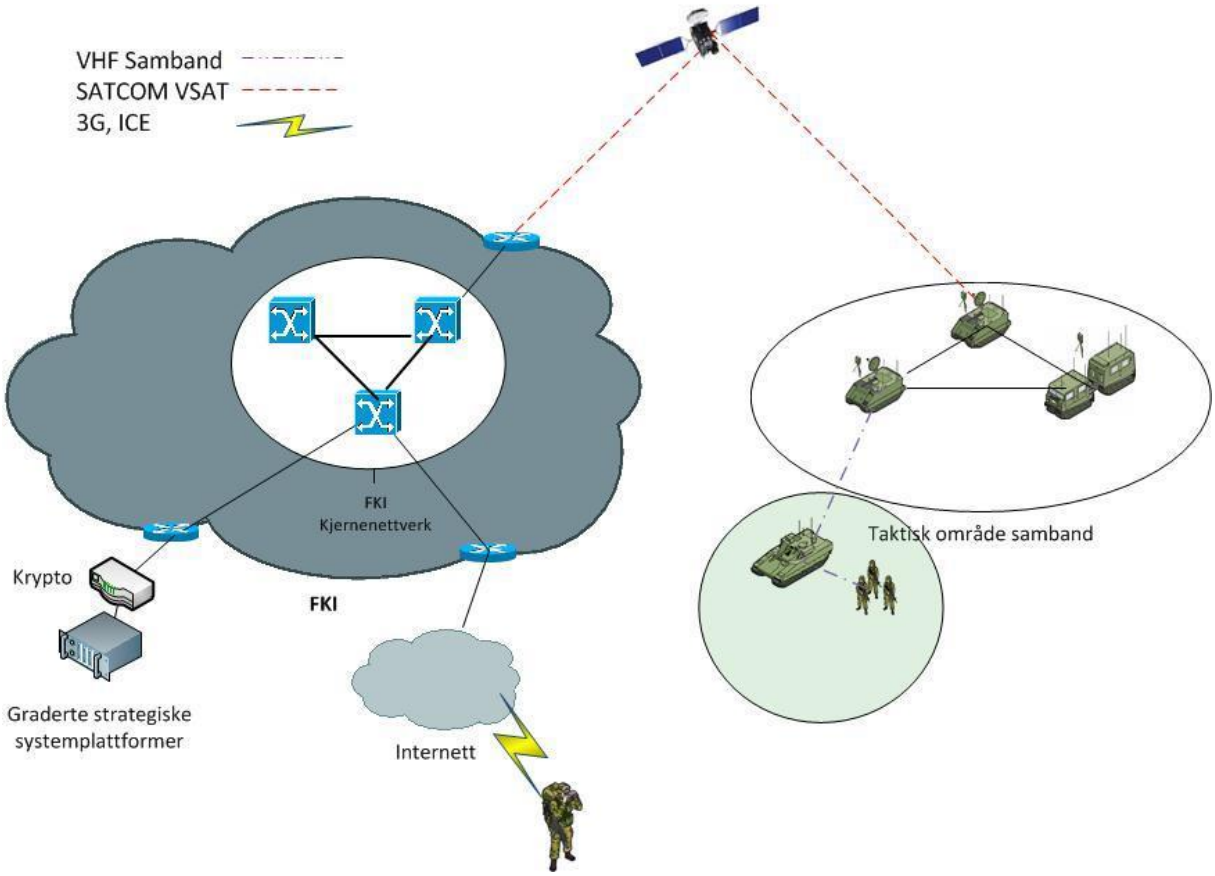
«Forsvarets kommunikasjonsinfrastruktur (FKI) produserer kommunikasjonstjenester for Forsvaret. FKI er systemer for nettverk og transmisjon samt hvordan disse systemene er koblet sammen for å produsere kommunikasjonstjenester [9]».

Ole Ingar Bentstuen (Fagråd FKI, 2009)

Gjennom litteraturstudie av oppgaven oppfatter jeg det slik at det er en felles konsensus mellom Forsvarsdepartementet (FD), Forsvarets forskningsinstitutt (FFI) og Cyberforsvaret (CYFOR) om spennområdet til FKI. Den inkluderer prosesser, informasjon, standarder og teknologi samt menneskene som drifter og vedlikeholder den og gjelder for både stasjonære, deployerbare og mobile elementer. FKI er en omfangsrik infrastruktur som er i kontinuerlig utviklingsprosess der nye løsninger fremskaffes basert på ny tilgjengelig og tilpasset sivil- og militær teknologi. FKI skal produsere felles standardiserte kjernetjenester som skal være tilgjengelig for hele Forsvaret.
FKI består av mange elementer og det er et omfattende tema. Vedlegg A gir en oppsummering på de viktigste prinsippene for å forstå FKI.

2.2.1 Kommunikasjon mellom mobile enheter og FKI.

Forsvarets nye taktiske områdesamband skal ha mulighet til å benytte forskjellige typer transmisjonsbærere, eksempelvis radiolinje, satellittkommunikasjon og Internett gjennom eksempelvis 3G. Og ved hjelp av disse bærerne skal det kunne etableres et områdedekkende nett. Systemet skal kunne operere autonomt, men kunne nyttiggjøre seg av kommersielle kommunikasjonstjenester. TOS vil sørge for å få tilgjengeliggjort tjenester fra den strategiske del av Forsvarets kommunikasjons infrastruktur til enheter i det taktiske domenet. Forbindelsen mot FKI skjer gjennom definerte termineringspunkter eller satellitt kommunikasjon (SATCOM) (figur 2).

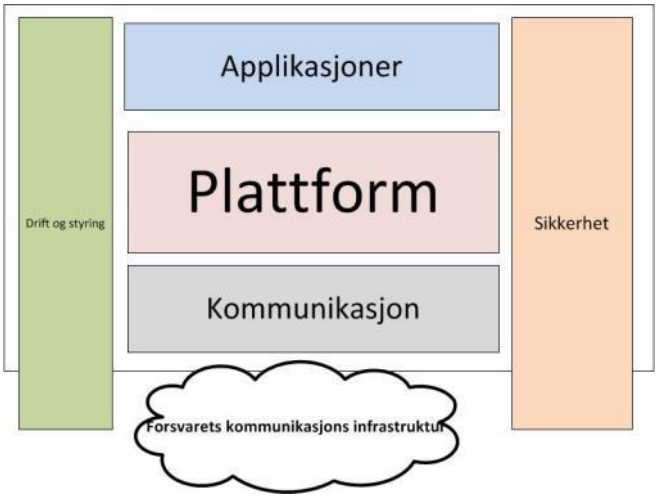


Figur 2: Forankring mellom mobile enheter og FKI.

Datatrafikk fra det taktiske områdedekkende sambandssystemet opp til det strategiske nivå forventes å øke betydelig. Innføringen av nye ulike sensorer på det taktiske nivå (strids teknisk), eksempelvis direkte overføring av videostrøm fra selve operasjonen til Forsvarets operative hovedkvarter (FOH) er et eksempel på dette.

2.2.2 Taktisk kommunikasjonsnode

Prosjektet Taktiske kommunikasjonsløsninger skal ta frem en kommunikasjonsløsning som går under navnet Taktisk kommunikasjonsnode (TKN). Prosjektet har i dag en pilotløsning som baserer seg på samme konfigurasjon som de øvrige mobile enhetene som er tilknyttet FKI.
Tjenestene som pilotløsningen gir er: Situasjonsbildet, meldingsfunksjonalitet og taletjenester. Prosjektet skal ta frem flere ulike TKN løsninger, fra bærbare og mobile løsninger satt i pansrede vogner til større flyttbare løsninger i konteinere. Noen viktige formål i prosjektet er innføringen av et kommunikasjons lag basert på IP-teknologi, etablere management for styring og overvåking av TOS og øke kommunikasjonssystemets kapasitet og fleksibilitet (figur 3).



Figur 3: Bruker plattform.

Kommunikasjonslaget i modellen er brukersystemets egne kommunikasjonstjenester som kan inneha de samme egenskapene som kjernenettverket i FKI. Kommunikasjonslaget har en egen forankring inn mot FKI.
Bærbare løsninger som for eksempel en soldat bærer med seg i felt eller mobile løsninger i et kjøretøy har båndbredde begrenset kapasitet. Det stilles derfor strenge krav til radio og nettverksutstyret som blir benyttet. Det må også være mulig å drive trafikkstyring og trafikkontroll av datatrafikken mellom strategisk nivå og taktisk. Sikkerhet må ivaretas i hele modellen. Derfor må kommunikasjonsløsningen sees i sammenheng med sikkerhetsløsningen slik at man oppnår nødvendig konfidensialitet, integritet og tilgjengelighet uten at det går på bekostning av tilgjengelig kapasitet.

De bærbare eller mobile løsningene skal også ha mulighet til knytte seg sammen *ad-hoc* og danne seg et autonomt *multi-hop* radio nettverk, også kalt et Mobile Ad-hoc Network (MANET), uten bruk av noen pre-eksisterende infrastruktur [10]. Mellomnoder i et MANET

kan fungere som rutere og videresende pakker på vegne av andre noder. På grunn av de spesielle egenskapene MANET har og nodenes mulighet til å takle raske endringer i topologien, gjør ad-hoc nettverk veldig attraktive for Forsvarets kommando, kontroll og informasjonssystemer (K2IS). Det er verdt å notere seg at ad-hoc nettverk introduserer ytterligere flere utfordringer og imperativer.

2.2.3 Management

I et TOS vil flere ulike taktiske kommunikasjonsnoder kommuniserer med hverandre og har tilknytning til FKI ved hjelp av ulike transmisjonsbærere. I prosjektet ønsker man å etablere management for styring og overvåking av trafikken som beveger seg mellom nodene og inn i kjernenettverket. Kjernenettverket må kunne gjenkjenne trafikken og gi datastrømmen nødvendige prioriteringer gjennom nettverket basert på operative krav. Oppstår det metning på en forbindelse eller at linken har for lav båndbredde kapasitet bør nettverket dynamisk kunne re-rute trafikken over andre forbindelser eller forkaste datapakker som ikke har prioritet. Det betyr at ulike tjenester må få ulike prioriteringer, datastrømmen må få nødvendig militær prioritet [11]. Det er et krav til Forsvarets kommunikasjonsinfrastruktur at datatrafikk fra tjenester som ikke er operasjonelt viktig må vike for trafikk som har militær prioritet, disse tjenestene må kunne fungere selv ved hundre prosent utnyttelse av kapasiteten i nettverket.
Som et resultat blir nettverksarkitekturen til Forsvaret komplekst og drift av en kompleks konfigurasjon blir ressurs intensiv og vanskelig å effektivisere.

De største utfordringene når det kommer til management kommer veldig ofte fra nettverksutstyret selv. Dagens nettverk er basert på IP teknologi for å identifisere og lokalisere servere, applikasjoner og tjenester. Dette går greit i et statisk nettverk av fysiske enheter, men er ekstremt arbeidskrevende for større virtuelle og dynamiske nettverk hvor trafikk variasjonene er store, nodene er mobile og båndbredden varierer.
Å gjennomføre drift og administrasjon av nettverk og miljø som stadig endrer seg ved hjelp av tradisjonelle midler har blitt ressurskrevende og dyrt å drifte. Dette gjelder spesielt virtuelle tilkoblinger, nettverkskonfigurasjon og mobile og bærbare enheter. For å forenkle oppgaven med å forvalte store fysiske og virtuelle nettverk, må administratorer bli frigjort fra bekymringene omkring den fysiske infrastrukturen som øker kompleksiteten.

2.3 Tjenestekvalitet og militær prioritet

2.3.1 Tjenestekvalitet

Tjenestekvalitet (QoS) kan være mekanismer for overføring av data på en måte som oppfyller bestemte krav eller det kan være et kvalitetsmål for tjenestene. Det finnes mange ulike definisjoner på tjenestekvalitet i faglitteraturen, et utdrag fra en definisjon er:

«QoS (Quality of Service) refers to a broad collection of networking technologies and techniques. The goal of QoS is to provide guarantees on the ability of a network to deliver predictable results (…)»[12].

I denne oppgaven vil jeg, basert på definisjonene, ha fokus på mekanismene for tjenestekvalitet.

Det eksisterer flere ulike strategier for å implementere tjenestekvalitet i et IP nettverk. En av de mest vanlige metodene er å bygge ut kapasiteten i nettverket i takt med behovet. Dette er på sikt ingen god løsning og «lik båndbredde til alle» er i praksis vanskelig å få til. En alternativ strategi er på en «intelligent» måte å fordele båndbredden som finnes mellom de ulike anvendelsene. To hovedkategorier går på å reservere ressurser eller å prioritere trafikk. Arkitekturene som er tatt frem er Integrated Services (IntServ) og Differentiated Services (DiffServ). Disse er beskrevet nærmere i vedlegg B.

2.3.2 Militær prioritet

Militær presedens av IP trafikk [13] kan defineres som viktigheten av trafikk for den militære bruker, organisasjonen eller systemet eller prosesser. Militær presedens må ikke forveksles med IP presedens [14]. En overordnet målsetning for IP tjenestekvalitet i nasjonale og NATO nettverk kan beskrives som følgende:
Muligheten for overføring av trafikk strømmer over forbindelser med full kontroll over QoS kravene i trafikk strømmene i nettverkselementene og i forbindelsene mellom nettverkene.

Videre kan følgende viktige krav stilles til IP tjenestekvaliteten:
Muligheten for en sømløs multi-level[1] prioritet på IP laget basert på viktigheten av trafikken. Når nettverket har stor belastning må den ha mulighet til å kunne endre policy eller iverksette minimum mode slik at viktig trafikk, militær trafikk, med høy prioritet fortsatt får tjenestekvalitet i motsetning til mindre viktig trafikk.

[1] Mulighet for å ha flere ulike køer med forskjellig prioritet for mange ulike typer trafikk.

Muligheten for en uniform nasjonal og NATO klassifisering og merking av trafikk. Konvertering (*mapping*) av datapakker på grensnittene mellom ulike nettverk, nasjonale domener eller mellom nasjoner innad i NATO kan være en opsjon.

Hovedutfordringene for å få dette til er beskrevet i vedlegg A avsnitt: *Elementer i FKI*, kryptosystemer og management på tvers av domener. Dette er vitale komponenter som må være på plass skal nasjoner og NATO kunne videreutvikle nettverksarkitekturen og for oppnå en mer skalerbar og robust QoS løsning. I tillegg er IP rutere og hardware teknologien en utfordring. Implementering av flere tjenesteklasser i en ruter er teknisk mulig, men samtidig å implementere en konfigurasjon for militær presedens er vanskelig å få til. Av [11] så har dette sammenheng med at en dedikert kø mekanisme må tildeles alle tjenesteklasser. Når metning oppleves i nettverket og tradisjonell kø- og forsinkelsesalgoritmer ikke kan løse problemet, må militær prioritet kunne bestemme hva slags trafikk som kan forkastes. Dette krever en detaljert bevissthet over trafikken i nettverket.

2.4 Vurdering av noen nettverksteknologier benyttet av Forsvaret

Forsvaret benytter seg av flere ulike typer nettverksteknologier i sitt kjernenettverk og på sine mobile og deployerbare plattformer: Asynchcronous Transfer Mode (ATM), Multiprotocol Label Switching (MPLS) og IP baserte nettverk (*best effort* baserte) er bare noen av teknologiene som er i bruk. I vedlegg C foreligger en kort beskrivelse av disse teknologiene og deres forhold til tjenestekvalitet.

2.5 Autentisering av tjenester og brukere

I et sikkerhetsperspektiv er det viktig for Forsvaret sin driftsorganisasjon å kunne identifisere brukere som ønsker tilgang til kommunikasjonstjenester i FKI, deretter gi korrekt autorisasjon og gjøre tjenesten tilgjengelig for brukeren. I tillegg må driftsorganisasjonen kunne gjennomføre effektiv styring og management av tjenesten og loggføre bruken. For konfigurasjonskontroll er det også viktig at brukere med rettigheter til å gjøre konfigurasjonsendringer på nettverkskomponenter blir også autentisert og endringer de gjør blir loggført. Disse sikkerhetsmekanismene i distribuerte systemer for å kontrollere hvilke brukere som gis tilgang til hvilke tjenester og spore hvilke ressurser de har brukt har fellesbetegnelse AAA-tjenester.
Autentisering (*authentication*), autorisasjon (*authorization*) og regnskap (*accounting*), er et rammeverk for kontroll av tilgang til ressurser, håndheve policyer, logging av bruk og metode for å innhente opplysninger for fakturering av bruk av tjenester i et nettverk. I

nettverk der effektiv drift og sikkerhet er en nødvendighet er disse kombinerte prosessene viktige.

Autentisering: Denne første prosessen identifiserer brukere typisk ved å bruke et gyldig brukernavn og passord. Autentiseringen er basert på at hver bruker har et unikt sett av kriterier for å få tilgang. AAA tjeneren sammenligner kriteriene som er lagret i en database og gir ikke brukeren tilgang til nettverkstjenestene hvis autentiseringsprosessen mislykkes.

Autorisasjon: Etter godkjenning må brukeren få autorisasjon for å kunne utføre oppgaver i nettverket. Autorisasjonsprosessen er å håndheve policyer: Hvilke tjenester får brukeren tilgang til, kvalitet på tjenestene, ressurser m.m. Vanligvis får man autorisasjon innenfor rammen av godkjenning.

Regnskap: Siste del av prosessen er regnskapsføring, som måler de ressursene brukeren forbruker i løpet av tilgangen. Det kan være hvor mye data bruker har sendt eller mottatt i nettverket, hvor lenge brukeren har brukt tjenestene, hvilke tjenester m.m. Regnskapet utfører logging av sesjonenes statistikk og informasjon om bruk av tjenestene til bruk for autorisasjonskontroll, fakturering, ressursutnyttelse og kapasitetsplanlegging i nettverket. Det er som regel en dedikert AAA tjener som utfører disse tjenestene. En mye brukt standard for nettverksaksess på grensesnitt er Remote Authentication Dial-In User Service (RADIUS) [15].

2.6 Oppsummering

Forsvaret sin kommunikasjonsinfrastruktur er et sammensatt nettverk bestående av mange ulike elementer. Nettverkene er heterogene, dynamiske og har store krav til kontroll. Forsvaret har mange like utfordringer som nettverksindustrien i sine kommunikasjonsnettverk. Men på grunn av Forsvaret sine avhengigheter til flere andre militære systemer og plattformer, NATO sin infrastruktur, strenge sikkerhetskrav og at kommunikasjonsforbindelsene i topologien har forskjellige kapasiteter gjør Forsvaret sine nettverk svært komplekse. For store og komplekse nettverk utgjør en sikkerhetsrisiko i seg selv, gode management løsninger blir derfor en nødvendighet for å oppnå effektiv og sikker samhandling mellom mobile, deployerbare og stasjonære plattformer.

I neste kapittel vil jeg presentere en av de største innovasjonene innenfor nettverksteknologier, Software Defined Networking.

3. SDN

Dette kapitlet gir en introduksjon til SDN og hvorfor SDN skiller seg ut i forhold til andre nettverksteknologier. Relevant forskning om SDN blir beskrevet i kapittel 4.

3.1 SDN

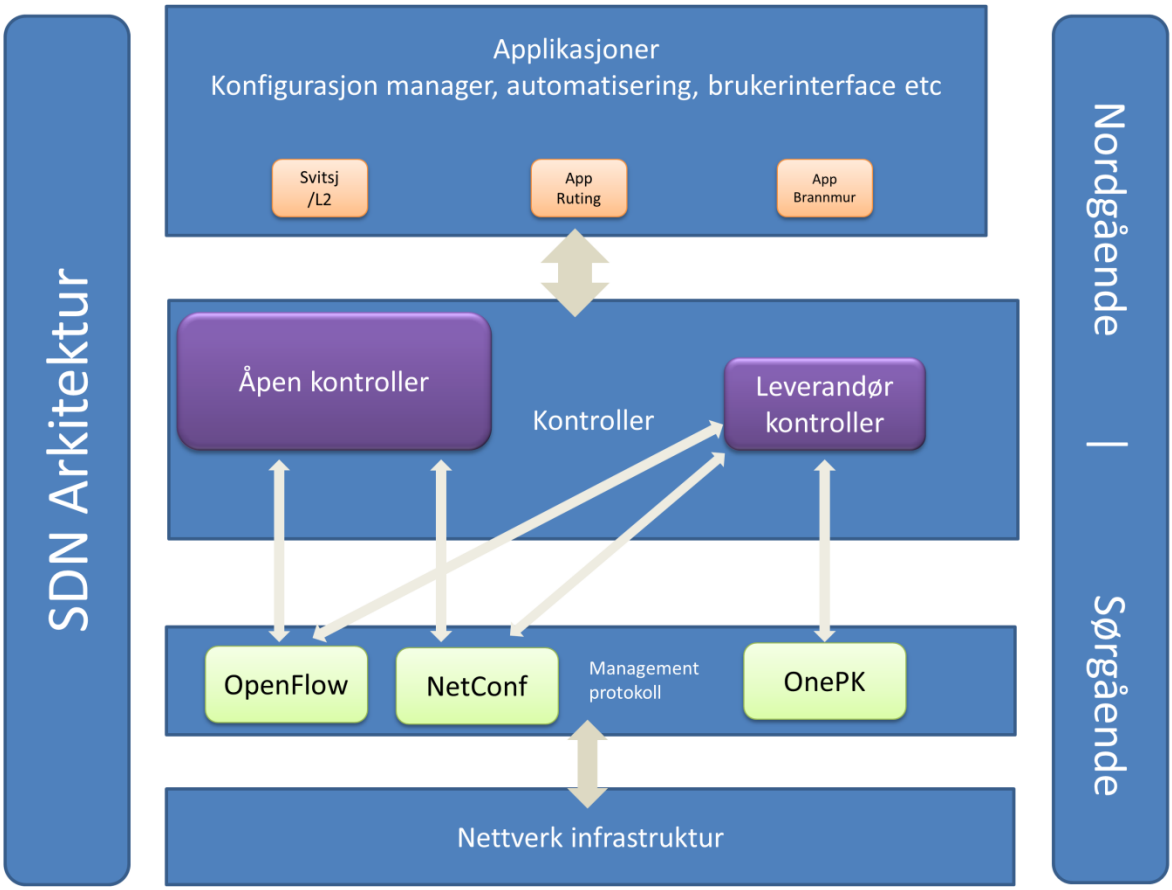
Nettverksutstyr som rutere og svitsjer blir tradisjonelt utviklet av leverandører med sin egen hardkodet programvare og annen programvare for å operere deres egen maskinvare. En standard ruter består av tre logiske plan: Kontroll planet, data planet og management planet, hvor kontroll og data planet har en tett kobling. Dette kan medføre til at implementasjon av nye tjenester, teknologi eller maskinvare i eksisterende nettverksinfrastrukturer blir vanskeligere.

«We believe that a number of important innovations are creating a paradigm shift in networking leading to higher levels of network programmability. These are: · separation of hardware from software; · availability of open programmable interfaces;(..)»[16].

I [16] blir programmerbare nettverk tatt frem som grunnlag for å revurdere dagens nettverksdesign gjennom tilgjengeligheten av åpen programmerbare grensesnitt og virtualisering av nettverk infrastruktur. SDN har de siste årene fått økt fokus ved akademier og i industrien.

Sofware-Defined Networking SDN is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today´s applications(..) (Open Networking Foundation) [17].

SDN er ikke en nettverksstandard, men et konsept innenfor nettverksarkitektur hvor kontroll- og data planet er adskilt fra hverandre. I figur 4 har jeg laget en illustrasjon som viser hovedelementene i en SDN arkitekturen.



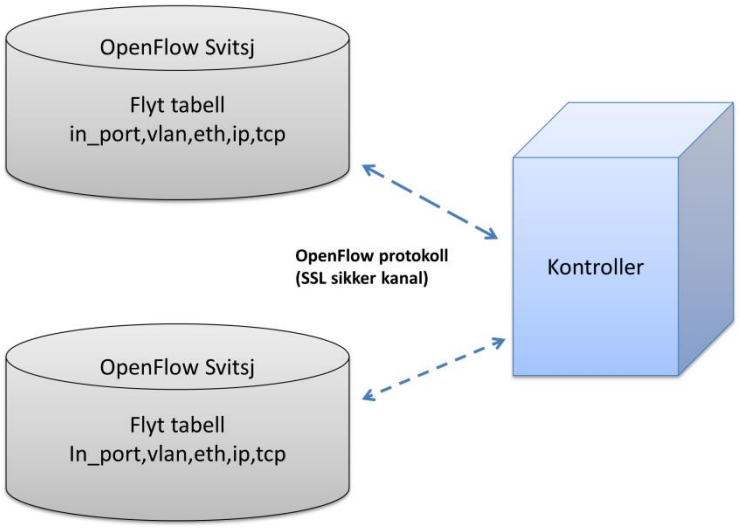
Figur 4: Hovedelementer i en SDN arkitektur.

Ved å oppdatere svitsje teknologien slik at data planet og kontroll planet kan bli separert, vil en programvarebasert sentralisert kontroller kunne etablere mer optimaliserte ruter til hensiktsmessig trafikk. Kontroll planet har oversikt over topologien i nettverket og genererer ruting tabeller, hvor data planet bruker tabellene fra kontroll planet og bestemmer hvor dataene skal sendes [18]. Denne seperasjonen gjør at nettverket kan være ytterligere abstrahert, noe som forenkler nettverksanalyse.
Kontroll planet kan være en tjener som kommuniserer med alle enhetene i nettverksinfrastrukturen ved hjelp av *Application Programming Interfaces* (API) og åpne standarder (OpenFlow) eller proprietære standarder (NetConf, onePK) kommunikasjons/managements -protokoller. En sentralisert kontroller betyr nødvendigvis ikke én kontroller. Logisk kan den være sentralisert, men eksisterer i en eller flere datasentre med redundans for kritisk infrastruktur. Kontroller logikken kan oppdateres gjennom programvare, noe som gjør det lettere å implementere ny funksjonalitet i nettverket hvis det er nødvendig.
Applikasjonslaget som vi finner øverst i lagdelingen (figur 4) omfatter de forskjellige APIs som eksisterer i nettverket for den enkeltes business strategi, eksempelvis: Video, tale tjenester, eller nettverkstjenester som ruting, tilgangskontroll og sikkerhetstjenester.

Applikasjonene kommuniserer med kontroll planet ved hjelp av åpne API standarder.[2] Nordgående representerer kommunikasjon mellom applikasjonslagene på toppen av komponenten, kontrolleren. Sørgående representerer kommunikasjon mellom kontroller og nettverksinfrastrukturen.

3.1.1 OpenFlow

OpenFlow (OF) er den første standarden designet for SDN [19]. Konseptet ble introdusert for første gang i en publikasjon av Nick McKeown, professor ved Stanford University [20] i 2008. Standarden var hovedsakelig designet for videreutvikling av campus nettverk, en metode for å kunne videreutvikle nettverket uten å måtte endre den underliggende maskinvaren i nettverket. OF separerer kontroll- fra dataplanet, dermed tar ut logikken fra nettverksutstyret. Det gjør det mulig å teste og verifisere nye nettverksprotokoller eller topologier uten å måtte gjøre endringer på underliggende nettverksutstyret. OF arkitekturen slik den først ble designet (figur 5):



Figur 5: OF arkitektur.

Hver svitsj eller annet nettverksutstyr i nettverket vedlikeholder en tabell for trafikkflyten (dataflyt tabell). Trafikk mottatt av utstyret blir videresendt basert på innholdet i tabellen. Konfigurasjonen er kontrollert av en OF kontroller. OF 1.0 spesifikasjonen definerer tolv

[2] På nåværende tidspunkt er det kun Cisco onePK som har standardisert grensesnittet mellom applikasjonslaget og kontroll planet ved hjelp av onePK SDK. OpenDaylight prosjektet har startet et samarbeid med Linux Foundation med ambisjoner om å gjøre det samme [18].

nøkkelfelt (figur 6) som en nettverksoperatør kan benytte for å lage metoder eller regelverk for dette formålet.

Ingress port	Ether source	Ether type	VLAN id	VLAN priority	IP src	IP dst	IP proto	IP ToS bits	TCP/UDP src/dst port	TCP/UDP Dst port

Figur 6: OpenFlow nøkkelfelt [21].

I OF arkitekturen er lag 2 egenskaper som for eksempel vedlikehold av *Media Access Control* (MAC) adresser, virtuelle nettverk (VLAN) eller lag 3 ruting protokoller som *Exterior Gateway Protocol* (EGP) eller *Interior Gateway Protocol* (IGP) implementert i kontrolleren i stedet for svitsjen. Kontrolleren genererer dataflyt tabellene for hver svitsj basert på disse protokollene og distribuerer dem. Ut i fra [22] kan kontrolleren distribuere dataflyt i strukturen ved hjelp av en reaktiv eller proaktiv tilnærming. Ved bruk av reaktiv distribusjon blir nye dataflyt implementert på nødvendige OF svitsjer i nettverket først når svitsj har mottatt første pakke i en dataflyt. Dette er en effektiv metode i forhold minne allokering i nettverksutstyret som skal vedlikeholde dataflyt tabellene. Ved proaktiv tilnærming distribuerer kontrolleren dataflyt tabeller til svitsjer basert på trafikk den mener kan komme.

3.1.2 SDN for heterogene nettverk

En kritisk egenskap for fremtidig nettverksarkitektur er støtte for heterogenitet, ref. Kapittel 1.1. SDN er foreslått som en metode for å kontrollere nettverk ved hjelp av programvare: Distribuere nye applikasjoner og tjenester samt justere nettverks policyer og ytelse. Målsetningene til SDN er hurtig utvikling og leveranse av ny arkitektur, standarder, programvare og applikasjoner, øke sikkerheten, stabiliteten og tilgjengeligheten i nettverket. SDN er selvorganiserende, i den grad nettverket kan selv velge det beste grensesnittet for overføring av trafikk basert på den totale topologi informasjonen kontrolleren har.

Men SDN sin foreløpige foreslåtte arkitektur baserer seg på sentraliserte kontrollmekanismer som egner seg ikke særlig bra for desentralisering, tidsavbrudd og forsinkelser som vi finner typisk i trådløse og mobile nettverk. Dette er viktige temaer som må drøftes og vurderes ved eventuelle implementasjoner av SDN for Taktiske kommunikasjonsløsninger.

3.1.3 Autentisering i SDN

Selv om SDN kan styrke håndhevelsen av policyer i nettverket må den underliggende arkitekturen være sikker og opprettholde prinsippene for konfidensialitet, integritet og tilgjengelighet. Selv i infrastrukturløse nettverk, eksempelvis MANET, der mobile noder (brukernoder) kan fungere som rutere for andre noder kan det være en utfordring å etablere sikre forbindelser ende til ende [23].
OF spesifikasjonen [19] beskriver bruken av sikkerhet i transport laget med gjensidig autentisering mellom kontrollere og mellom kontroller og svitsj. Men sikkerhetsegenskapen er valgfri. Autentisering i SDN blir diskutert nærmere i kapittel 4.1.4 og 5.

3.1.4 Håndtering av tjenestekvalitet

I kapittel 2.3 ble tjenestekvalitet og militær prioritet beskrevet, i tillegg til løsninger for håndtering av tjenestekvalitet i tradisjonelle nettverksarkitekturer (ref. Vedlegg B). Ved hjelp av dekoblingen i SDN vil kontroll planet ha den logiske oversikten av hele nettverket og ville kunne håndtere tjenestekvalitet der det måtte være behov for i nettverket ved hjelp av distribusjon av policyer, *slicing* eller skalering. For OF standarden ble tre QoS primitiver foreslått [24, 25]:

Rate promising or Minimum datarate	*Slicing*: Gir hver dataflyt (*flow*) minimum båndbredde garanti på hver egress[3]node. Hvis noden ikke har mulighet til å gi egress hastighet pr dataflyt, blir trafikken merket med en *Rate Promise ID* og lagt i tilhørende kø til sin tjenesteklasse. Kontrolleren har mulighet til å sette opp og konfigurere køer og *mappe* hver dataflyt til den spesifikke kø. *Slicing* mekanismen er delt i to: **Konfigurasjon:** Hvordan køene blir konfigurert er forskjellige fra kontroller til kontroller. Dette er ikke spesifisert i OpenFlow standarden. ***Flow Queue Mapping/Forwarding*:** Involverer å forflytte trafikk fra kø til korresponderende handling.
Rate Limiting or Maximum Rate	*Rate limiter* kontrollerer hastigheten av dataflyten og er delt i to: **Egress,** regulerer eventuelt øvre grense på hastigheten av overføringen av en dataflyt på et egress grensesnitt på noden. **Ingress**[4], også kjent som *policing*. Ingress fjerner datapakker hvis de ankommer raskere enn en predefinert grense på ingress grensesnittet på noden.
Strict precedence	Denne egenskapen gir prioritets nivå til dataflyt ved hjelp av *Virtual Local Area Network Priority Code* (VLAN PCP)[5] eller *Type Of Service* (TOS)[6] verdien i IP hodet [26, 27].

Tabell 1: OF QoS primitiver.

Open Network Foundation[7] [28] antar at en fremtidig versjon av OpenFlow standarden (2.0)[24] vil ha mulighet til å tildele QoS på dataflyt. Standarden må derimot ikke bli for kompleks og ta bort en av sine største fordeler som nettopp er å være enkel.

[3] Egress node er en ruter eller tilsvarende hvor datapakker forlater nettverket til et annet nettverk.
[4] Ingress node er en ruter eller tilsvarende hvor datapakker ankommer nettverket fra et annet nettverk.
[5] IEEE 802.1Q
[6] RFC1349
[7] ONF er en industri konsortium som aktivt støtter SDN utviklingen og er ansvarlig for OF standarden.

3.2 Oppsummering SDN

SDN har sin opprinnelse fra OF prosjektet og var initielt en teknologi med sentralisert kontroller og seperasjon av kontroll og data planet. OF ble benyttet til å kommunisere til dataplanet. Dette var mekanismer i SDN, i dag er SDN et rammeverk i en nettverksinfrastruktur for å løse mange ulike problemstillinger. SDN gir mulighet for virtualisering, mulighet for kontroll og styring av tusenvis av enheter ved hjelp av en eller få kommandoer, den er programmerbar og kan automatisere mange driftsoppgaver i et nettverk for å nevne noen av egenskapene.

meldinger: 1) «pakke-inn» melding fra OF svitsj til kontroller og 2) pakke-ut eller endrings melding fra kontroller som respons fra 1). Selv når forbindelsen mellom svitsj og kontroller, OF kanalen, har tilgjengelig båndbredde og kontrolleren er rask, eller etterspørselen av oppsett av dataflyt er lav til moderat, viser analyse av målinger at *round-trip time* (RTT) [8], kan være vesentlig lang.
Videre ble kontrollerens responstid testet når antall forespørsler fra svitsjene økte.
Resultat av testene viser at når svitsjen må regelmessig sende forespørsler til kontrolleren for hvordan de skal behandle pakkene, som medfører til økt prosesseringstid på kontrolleren og kunne gi tendens til metning på OF kanalen, gir en alvorlig flaskehals for skalerbarheten til OF. Kontrollerne Nox, Nox-MT, Beacon og Maestro ble testet i [32].

SDN for heterogene nettverksmiljø blir diskutert i [33], fokuset er spesielt infrastrukturløse nettverk: Mobile enheter med begrenset eller periodisk konnektivitet til nettverksinfrastruktur ved hjelp av trådløs teknologi eller mobilnett og har mulighet til å forme ad-hoc nettverk. Flere aspekter av SDN blir diskutert og en rekke krav og utfordringer blir skissert, blant annet:
i) Begrensningene som eksisterer på mobile (håndholdte) enheter i forhold til strøm, ytelse på prosessor, minne og kommunikasjon.
ii) Hvordan kontroll planet skal håndtere enheter som ikke er SDN kompatible og bruken av distribuerte kontroll plan.
Videre diskuterer publikasjonen hvordan SDN sørger for at kontroll meldinger når nodene sine kontrollert og sikkert. Forfatterne av publikasjonen påpeker at SDN har mangelfulle mekanismer for dobbel autentisering for å sikre legitimiteten av noder og kontrollere, i tillegg til finnes det ingen mekanismer som sjekker ektheten av kontrolltrafikk og at enheten fungerer som forventet i respons til instruksjoner.
Ivaretakelse av sikkerheten å garantere konfidensialitet, integritet og tilgjengelighet i heterogene nettverk er en utfordring i seg selv, for *Mobile Ad-hoc Networks* (MANET) [34] blir også problemstillingene omkring *peer-to-peer* arkitekturen og *multi-hop* ruting diskutert.

Andre publikasjoner som omhandler SDN rammeverket, nettverksdesign og programutvikling er oppsummert i tabell 2:

[8] RTT defineres som intervallet mellom forekomsten når pakke-in melding er sendt fra svitsjen og forekomsten når en pakke-ut eller endring melding er mottatt.

Referanse	Innhold	Foreslått løsning eller anbefaling
[35]	Publikasjonen drøfter utfordringer med OpenFlow (v1.2) i produksjonsnettverk.	Utvidelse av OpenFlow på FlowTable, kontrollmodus og OpenFlow protokollen
[36]	Publikasjonen er en kartlegging og veiledning av varierte studier av SDN og arkitekturen. Videre kartlegger den initiativer fra ulike forskningsmiljø som har målsetning om å løse en del identifiserte problemstillinger med SDN, eksempelvis kompabilitet og interoperabilitet av OF, som er relevante i dagens produksjonsnettverk.	
[37]	Publikasjonen diskuterer problemstillingene omkring kompleksiteten til SDN i et programutviklingsperspektiv. Diskusjonen omhandler: Hvordan gjøre SDN mer effektivt, hvordan optimalisere den for alle typer nettverksløsninger og definere avveininger mellom ulike implementasjoner? I markedet eksisterer det flere applikasjoner og kommersielle produkter av SDN, hvordan kvantifisere og teste ytelsen av dem?	Behov for å ha en kvantitativ tilnærming for å evaluere ytelse og effektivitet av SDN. Flere faktorer som medfører til at SDN design er komplisert: 1) Variasjon i topologien. 2) Ytelse. Det må forskes mer på relevante beregningsmetoder for å evaluere ytelsen til SDN. 3) Beregnings kompleksitet. Det er flere faktorer med SDN som må tas i betraktning: Forsinkelser og metning på de ulike OF kanalene og tilgjengelige ressurser. 4) SDN design. Implementasjonen må kunne være skalerbart og tolerant for feil på tvers av flere noder.

Tabell 2: Oversikt over publikasjoner - SDN konsept og design.

4.1.3 Hybride nettverksløsninger med OpenFlow/SDN

Den hybride arbeidsgruppen i ONF [28] har et spesielt fokus på hybrid svitsjearkitektur. Målsetningen er å ha nettverkskomponenter som kan bli konfigurert til å fungere seg som en ordinær svitsj eller som en OF svitsj og i noen tilfeller som begge deler. I [38] blir motivasjonen til arbeidsgruppen og en del problemstillinger omkring hybride nettverksmodeller i OF(v1.2) og SDN introdusert. Publikasjonen presenterer også to kontroller-sentriske tilnærminger for å kombinere tradisjonelle kontroll plan funksjoner med OF: LegacyFlow [39] og RouteFlow [40].
Publikasjonen drøfter også noen bekymringer for innføring av hybride nettverksløsninger i eksisterende nettverk, ettersom den berører både økonomiske og tekniske problemstillinger, spesielt kompatibilitet av OF på ordinært nettverksutstyr og på tvers av ulike domener.

4.1.4 Tjenestekvalitet i OpenFlow/SDN

I [25] blir flere problemstillinger omkring OF standarden og dets håndtering av tjenestekvalitet drøftet. OF er en relativt ny protokoll og er fortsatt under utvikling, det eksisterer derfor en del utfordringer i OF standarden om den skal kunne håndtere QoS effektivt. Det er essensielt for OF å ha noe grunnleggende funksjonalitet for QoS, samtidig, for mye funksjonalitet vil gjøre OF protokollen for kompleks og gi økt prosesseringstid for svitsjer og rutere.
Det finnes ikke støtte for QoS i tidligere versjon av OF(1.0-1.2), nyere versjoner har muligheter til å tildele QoS på pakker og rapportere til kontrolleren QoS kapasitetene den støtter. Det heller ingen garanti for at maskinvaren i svitsjen støtter nyere versjon av OF og kan medføre at enheten slutter å fungere som normalt (ref. Kapittel 4.1.2).
Eksisterende versjon av OF baserte svitsjer kan kun kontrollere DiffServ klasse basert trafikk. En QoS-kontroller er blitt introdusert til rammeverket [41], med ansvar å allokere ressurser til QoS sensitive dataflyt automatisk for å garantere minimum båndbredde og minimum forsinkelse. QoS kontrolleren får som hovedoppgave å reservere nettverksressurser slik at Service Level Agreement (SLA)[9] krav kan bli overholdt.

Multimedia tjenester, for eksempel videostrømmer ved bruk av SDN og OF blir drøftet i [42]. Multimedia tjenester har strenge krav til pakke tap, forsinkelser og jitter. God QoS implementasjon skal kunne minimere disse problemstillingene. Ved hjelp av OF kan kontrollere overvåke dataflyt og innhente statistikk, deretter ved hjelp av ruting algoritmer beregne minimum kost på ruter og opprettholde spesifikke krav til tjenesten i forhold til variasjon av forsinkelse fra kilde til destinasjon i nettverket.
En oversikt over forskjellige forslag på QoS-orientert design med bruk av OF blir presentert

[9] SLA er en tjenestekontrakt mellom tjenesteleverandør og bruker hvor tjenestene er definert og spesifiserer blant annet: Omfang, kvalitet, ansvar m.m.

og diskutert i [43], og viser forbedringer som kan implementeres i en fremtidig OF standard eller forbedringer i SDN konseptet. Studien påpeker at utvidet testing er nødvendig for ulike kontroller design og dynamisk ruting algoritmer.

4.1.5 Sikkerhet i SDN

Kontrolleren er hjernen i SDN, uten skikkelig sikkerhet rundt kontrolleren blir nettverket sårbart for uheldige endringer eller ulike angrep. Publikasjonene [44] og [45] presenterer en omfattende undersøkelse av forskningen knyttet til sikkerhet i SDN. Både sikkerhetsforbedringer og utfordringer i SDN rammeverket blir diskutert. SDN kan gi betydelige forbedringer i nettverkssikkerheten ved å utnytte potensialet som ligger i programmerbarheten og sentralisering av en del nettverksfunksjonalitet. Samtidig eksponeres nettverket for en rekke nye sikkerhetsutfordringer, trusler og potensielle angrep, eksempelvis *Denial-of –Service* (DoS), på grunn av sentralisert kontroller og begrensninger av dataflyt tabeller i nettverksenhetene.
En annen problemstilling er «tillit» mellom applikasjoner og kontroller, og kontroller og nettverksenheter som blir i [33] drøftet også for mobile enheter.
I [46] blir OpenFlow spesifikasjonens bruk av Transport Layer Security (TLS) for autentisering mellom kontroller og svitsjene drøftet. Forfatterne påpekte at sikkerhetsegenskaper i SDN er valgfrie og at standarden for TLS er ikke spesifisert. Mangelen på innføring av TLS hos store leverandører gjør OF utsatt for *man-in-the-middle* angrep [47], uredelig regel innsetting og modifisering.

4.1.6 Relevant forsvarsprosjekt

Tactical Communications, Communications Standards for Federated Networking (TACOMS+), er et multinasjonalt prosjekt som jobber utenfor NATO strukturen men blir finansiert fra nasjonene. Visjonen til prosjektet er å opprettholde og fremme *Standard NATO Agreements* som gjør det mulig for landene å oppnå interoperabilitet mellom nasjonale kommunikasjonsnettverk under militære operasjoner.
TACOMS STANAG definerer et sett av universelle grensesnitt, IOP*,* som kan monteres i de fleste eksisterende nettverkssystemer eventuelt planlegges i fremtidig infrastruktur. IOP er et «bindeledd» mellom føderasjoner av heterogene nettverk.
På grunn av at dagens operasjoner i NATO er langt mer mobile og preget av hurtig reaksjonstid fra deltakernasjonene, er operasjonssjefer avhengig av nettverksressurser som er enkel å distribuere og bruke. De må tillate fleksibilitet med hensyn til nettverkstopologien og tillate rask tilpasning i forhold til operasjonsegenskaper eller geografi. Utplassering av dagens løsninger krever mye manuell konfigurasjon. Dette er både arbeidsintensiv og tidkrevende, noe som øker risikoen for menneskelige feil i systemkonfigurasjonen.
Et mål for TACOMS+ prosjektet er å utvikle automatiserte teknikker for konfigurasjon av

nettverket og nettverkslaget og støttende nettverksfunksjoner som vil føre til en raskere og kostnadseffektiv distribusjon av en FoN infrastruktur.
Høsten 2014 ble en løsning for automatisk ruting distribusjon mellom nasjoner ved hjelp av Cisco sin SDN løsning, onePK, presentert. Dette var en av flere arbeidspakker.

4.1.7 Oppsummering

Dette kapittelet har gitt et sammendrag av publikasjoner som kan være relevant for å forstå SDN konseptet og som er aktuelt for denne oppgaven.
I neste kapittel blir scenarioer og løsningsdesign med bruk av SDN for å understøtte Forsvarets taktiske kommunikasjonsnoder beskrevet.

5. SDN for Taktiske kommunikasjonsnoder

For å få en bedre forståelse av problemstillingen i denne oppgaven (ref. Kapittel 1) må SDN konseptet drøftes mot Forsvaret sin eksisterende infrastruktur (ref. Vedlegg A). Kan SDN benyttes for løse disse utfordringene? I så fall, hvilke design vil være mest hensiktsmessig? De funksjonelle kravene til TOS (ref. Kapittel 2 og Vedlegg A) er beskrevet i et kravdokument, valg av SDN teknologi og løsning må vurderes opp mot disse.
Dette kapittelet ser nærmere på SDN i FKI, deretter blir problemstillingene for taktiske kommunikasjonsnoder presentert i to ulike scenarioer. For hvert scenario blir mulig design ved hjelp av SDN presentert. Metoden for å kunne evaluere en løsning blir diskutert i kapittel 6, deretter følger implementasjonen av løsning i kapittel 7.

5.1 Effektivisering av drift

Prosjektet Taktiske kommunikasjonsløsninger skal ta frem flere TKN løsninger (ref. Kapittel 2.1.1). Et av prosjektets formål er å etablere management for *styring* og overvåking av TOS. Styring kan være *etablering av policyer* på grensesnitt for å kunne *styre prioritert trafikk* i nettverket eller overvåkning av strategisk viktige noder i en operasjon. Styring kan også være *regulering av hvem som får tilgang til nettverksressursene*, hva de får lov å gjøre og overvåkning av aktiviteten. Feil i konfigurasjonen på en entitet i nettverket kan ha store ringvirkninger på hele nettverket. Forsvaret sine heterogene nettverk kjennetegnes blant annet med regelmessige endringer i nettverkstopologien. Nodene har ulike bærere med forankring til FKI og har forskjellige båndbredde kapasiteter. For å styrke sikkerheten i slike nettverk må navn (brukere) og brukersystemer være fast bundet til entydige innretninger, plattformer og porter (ref. Kapittel 2.2.2). Management av Forsvaret sine nettverk er i dag preget av manuelle rutiner og vedlikehold. For hver operasjon gjennomføres det omfattende sambandsplanlegging og manuell konfigurasjon av systemplattformer og nettverkskomponenter. Kan SDN effektivisere noen av disse utfordringene og automatisere enkelte driftsrutiner og drive styring?

5.2 Programmerbart - ny funksjonalitet

Programmerbarheten i SDN betyr at løsninger for management og styring kan skreddersys kravene til Forsvaret sine nettverk innenfor de rammene som foreløpig er mulig. SDN er et konsept og ikke én teknologi eller nettverksløsning, de tekniske kravene til et programmerbart nettverk (SDN) må derfor skisseres og sees opp mot de funksjonelle kravene i FKI og TOS som er utarbeidet fra brukerens krav til tjenestene. Problemstillingene kan derfor deles opp i tre ulike krav: Brukerkrav, funksjonelle krav og tekniske krav.

Brukerkrav: En bruker på en TKN skal kunne få tilgang til kjernetjenester i FKI.
Funksjonelle krav: Løsning skal kunne automatisk autentisere bruker, autorisere og gi tilgang.
Tekniske krav: Brukergrensesnitt (spesifikke detaljer), integritet og konfidensialitets beskyttelse ved hjelp av for eksempel kryptering og krav til kommunikasjon.

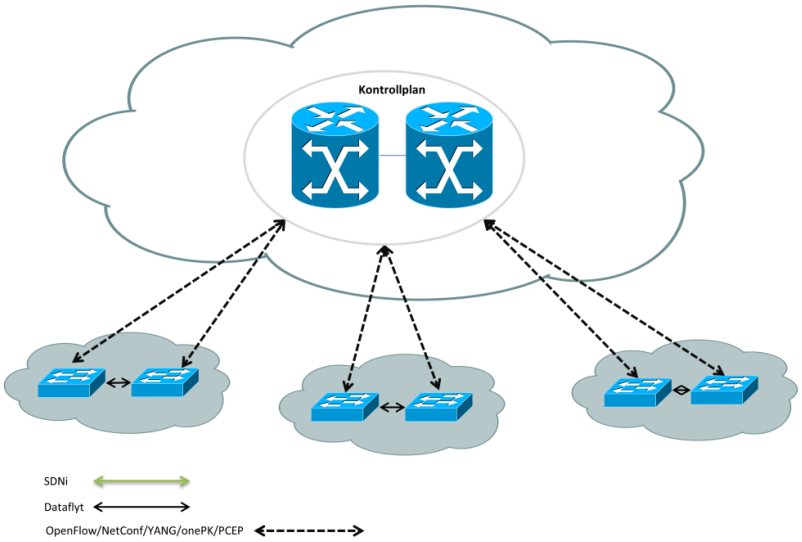
Dette var en veldig overordnet beskrivelse av prosessen for implementasjon av nye tekniske løsninger i et operasjonelt nettverk. Prosessene i et IKT investerings- og utviklingsprosjekt er et omfattende og omstendelig arbeid.

Programmerbarheten kan medføre til økt kompleksitet (ref. Kapittel 4.1.2, tabell 2), skal SDN kunne benyttes til understøtte de funksjonelle kravene i problemstillingene må SDN optimalisere eksisterende nettverksløsning og effektivisere FKI. Implementering av nye sikkerhetsfunksjoner ved hjelp av SDN må samtidig ikke gi nye sikkerhetsutfordringer i nettverket (ref. Kapittel 4.1.4).
Basert på min litteraturstudie av Forsvaret sine nettverk og SDN kan etter min vurdering, problemstillingene lettere drøftes ved hjelp av faktiske scenarioer for TKN. Scenarioene blir beskrevet i kapittel 5.4 og 5.5

5.3 SDN I FKI

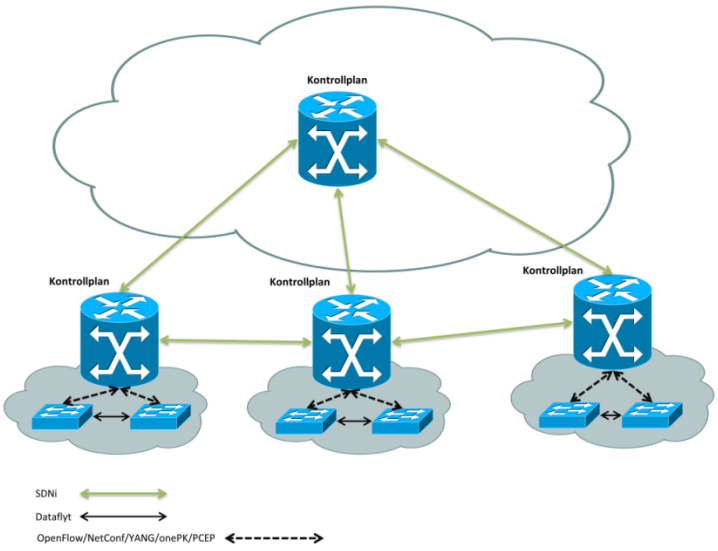
SDN kontrolleren skal blant annet kunne regulere tilgangene og drive styring. En utfordring med arkitekturen er plasseringen av SDN kontrolleren i FKI for å drive management og styring. Følgende forslag til løsninger kan benyttes:

- Sentralisert kontroller i kjernen av FKI (figur 6).



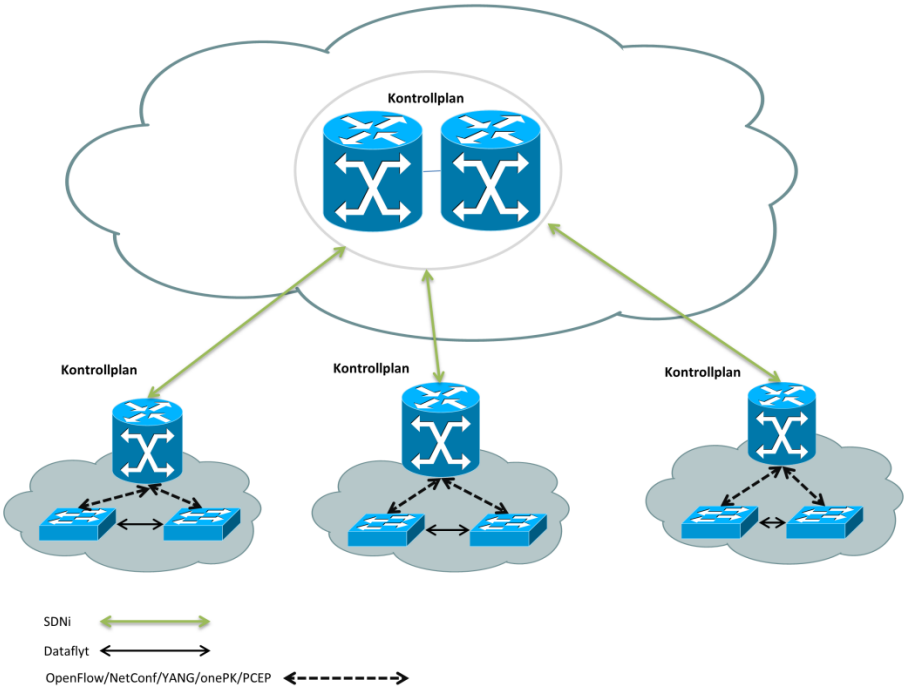
Figur 6: Sentralisert kontroller arkitektur.

- En kontroller for hvert brukersystem, distribuert infrastruktur (figur 7).



Figur 7: Distribuert kontroller arkitektur.

- Kontroller hierarki, hver kontroller får ansvar for hvert sitt domene (figur 8).



Figur 8: Hierarkisk kontroller arkitektur.

Med én **sentralisert kontroller** (figur 6) vil kontrolleren til enhver tid ha oppdatert topologioversikt av nettverket, så fremt at alle brukersystemene i nettverket har kommunikasjon til kontrolleren.
Men selv med redundans i FKI kjernen ved hjelp av cluster løsninger vil dette være en mindre hensiktsmessig løsning på grunn av at kontrolleren blir veldig utsatt i nettverket, dette betegnes som «single point of failure»[10]. Hele nettverket blir påvirket hvis kontrolleren skulle slutte å fungere.
I tillegg, med ulike kapasiteter til brukersystemene vil forsinkelser i kommunikasjonen forekomme, dette kan påvirke implementasjonen av nye policyer og dataflyt tabeller på svitsjene i brukersystemene (ref. Kapittel 4.1.2). En sentralisert kontroller i FKI må også kunne håndtere et stort antall dataflyter. Oppsett av mange dataflyter vil trolig gi for store beregningsbelastninger for en kontroller til å håndtere effektivt [6].
Kontrolleren er også utsatt for Denial of Service (DoS)[11] angrep som er en sårbarhet i SDN plattformen og kan ha ødeleggende effekt på hele nettverket (ref. Kapittel 4.1.5).
Men hva med management grensesnittet, har det båndbredde krav? Hva med

[10] *Single point of failure*: Kjennetegnes i en stjernetopologi. Hvis den sentrale entiteten slutter å fungere påvirkes hele nettverket.
[11] DoS angrep er et tjenestenektangrep som vil føre til et brudd på tilgjengeligheten til informasjonen eller ressursen.

brukersystemene som i lengre perioder ikke har kommunikasjon med FKI?
Sentralisert kontroller arkitektur vurderer jeg som en urealistisk løsning for FKI.

I en **distribuert kontroller arkitektur** (figur 7) med en kontroller for hvert brukersystem, vil kontrollere være avhengig av å kunne kommunisere med hverandre.
SDNi [44] er foreslått som standard for inter-kontroller kommunikasjon. I denne infrastrukturen finnes det ikke en sentralisert kontroller med den totale topologioversikten. Kontrollerne må derfor sende sin topologi oversikt til nabo kontrollere for hver gang det skjer en endring i egen topologi. SDNi protokollen eksisterer kun som et utkast til en standard. Derimot kan leverandører av SDN løsninger utvikle egne protokoller for å håndtere kommunikasjonen mellom sine egne kontrollere. Men er det hensiktsmessig at hver kontroller i infrastrukturen skal ha den totale topologioversikten?

Problematikken omkring beregningsbelastninger og forsinkelser med bruk av en sentralisert kontroller arkitektur vil oppstå også her. Kontrollerne må kalkulere store tabeller som medfører til mye management trafikk i nettverket. En bedre løsning kan være at kontrollerne har kun oversikt over sitt brukersystem og sine kommunikasjonsmuligheter til FKI og setter opp dataflyt basert på forespørsler fra andre kontrollere. Mekanismer i kontrollerne må også kunne håndtere duplisering eller feil i topologi informasjonen som er gitt.

En **hierarkisk oppbygging av kontrollere** (figur 8) vil en sentral kontroller i FKI kjernen ha den totale topologioversikten. Kontrolleren kan motta og sende topologi informasjon til andre kontrollere i aksesspunktene til brukersystemene. Den sentrale kontrolleren kan kalkulere ruter i nettverket basert på informasjonen den får og distribuere dataflyt og policyer til underliggende kontrollere. Igjen, en sentral kontroller betyr logisk én, men kan bestå av flere kontrollere i et datasenter.

I forhold til last balansering og redundans vurderer jeg denne arkitekturen som den beste løsningen for FKI. Sentralisere kontroll planet i en logisk sentralisert men fysisk distribuert modell er mer hensiktsmessig for skalerings-, tilgjengelighets- og geografiske perspektiver. Forskjellig båndbredde kapasiteter ut til forskjellige domener og brukersystemer i FKI vil fortsatt kunne gi forsinkelser og skape utfordringer for styring og management [6]. Sårbarhetene i SDN plattformen kan også gjøre nettverket utsatt for eventuelle angrep [7][28-29].

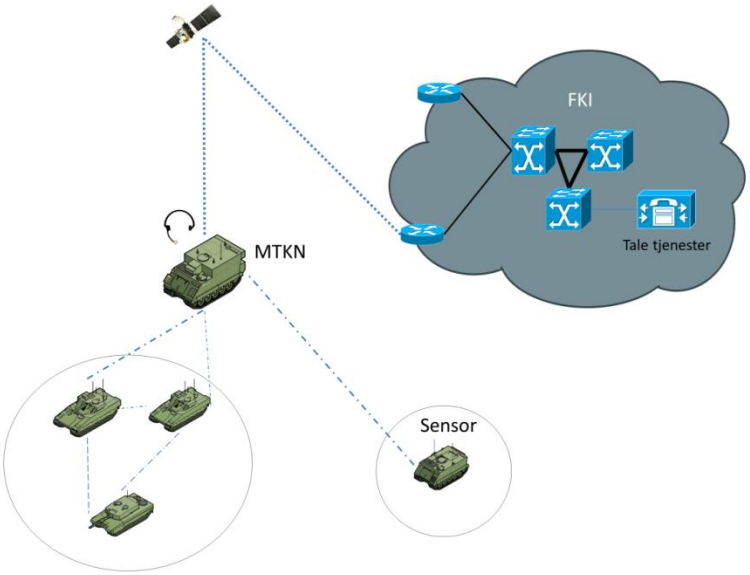
5.4 AAA tjenester for TKN i FKI

AAA designet er å forsikre om at kanalen mellom applikasjon, kontroller og dataplanet er sikker og kunne kontrollere ressursene brukere har tilgang til og hva de gjør i nettverket. Kontroller må kunne registrere og autentisere svitsjer, autentisere noder og binde de til svitsjer og eventuelt porter, autentisering av bruker og autorisasjon, tilslutt autentisering av tjenestene.

5.4.1 Scenario: Tilgang til kjernetjenester for TKN

Gjennom et planarbeid før militære operasjoner skal de kvalitative og kvantitative system karakteristikk være beskrevet i en *Concept of operations* (CONOPS)[48]. Dette er en beskrivelse av primær og sekundær oppgaver, kapasitetene til kommunikasjonssystemene (samband) og hvordan et sett med funksjoner kan benyttes for å oppnå ønsket sluttilstand. Dagens nettverksdesign har derfor et statisk oppsett der aksessen og tjenestene MTKN ønsker å benytte fra FKI er definert på forhånd og manuelt programmert i rutere og svitsjer. Tilgang til tjenester av prinsippet «ad-hoc» er vanskelig å få til.

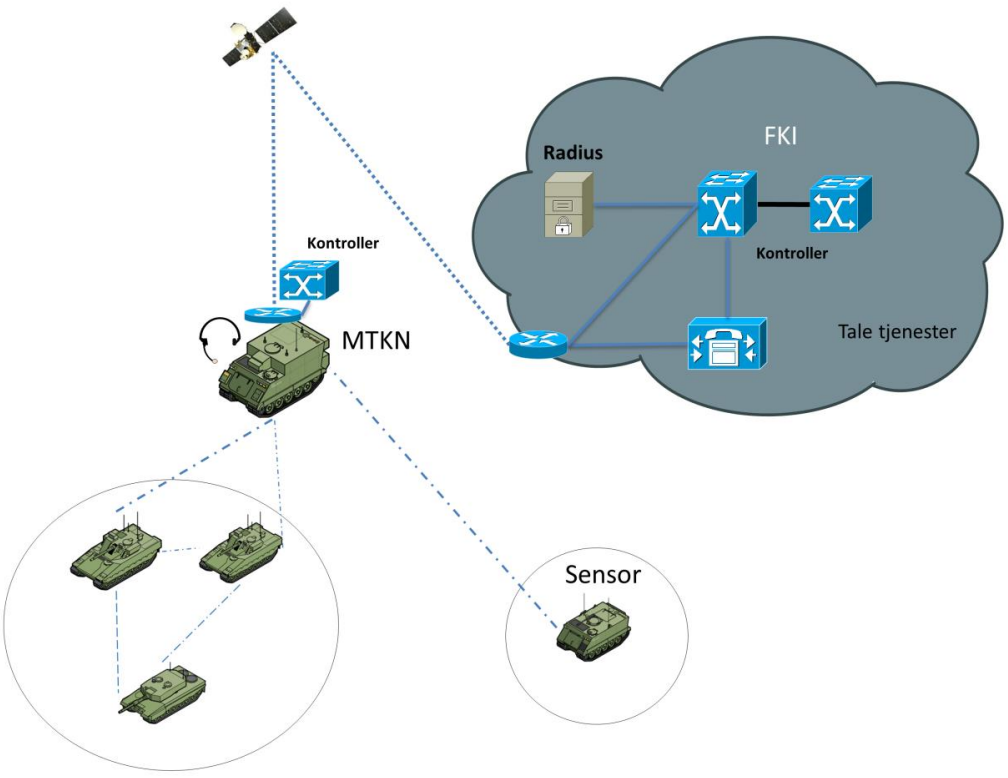
Figur 9 illustrerer en MTKN (middels stor taktisk kommunikasjonsnode) som ønsker tilgang til kommunikasjonstjenester i FKI, i dette scenarioet telefoni. Noden har en plattform med egne brukersystemer der integritet og konfidensialitet er ivaretatt av kryptosystemer (ref. Vedlegg A). MTKN har et aksesspunkt gjennom en link til FKI ved hjelp av SATCOM. MTKN fungerer i dette tilfelle som en IOP for de andre nodene og tilgjengeliggjør tjenester i FKI for eksempelvis sensor-kjøretøyet.



Figur 9: TKN med behov for kjernetjenester.

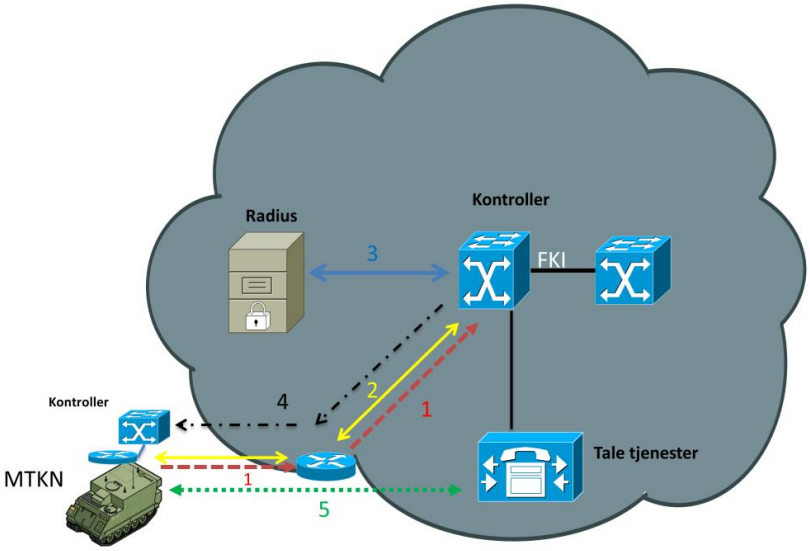
5.4.2 Design med bruk SDN

SDN kan ivareta sikker kommunikasjon mellom kontroller og nettverksenhetene. SDN har derimot mangelfull støtte for å håndtere tillitsparadigmer på nordgående og sørgående grensesnitt (ref. Kapittel 4.1.5). På lik linje med SDNi er ikke grensesnittet mellom applikasjon og kontroller standardisert ennå (ref. Kapittel 3). Leverandører av SDN teknologi har egne løsninger for sine kontrollere, disse SDN teknologiene må vurderes for denne oppgaven.
For at kontroller skal kunne håndtere autentisering, autorisasjon og regnskap i nettverket må ytterligere funksjonalitet implementeres. Figur 10 skisserer et konsept for design ved hjelp av SDN, hierarkisk kontroller arkitektur (ref. Kapittel 5.3) og *Remote Authentication Dial In User Service* (RADIUS) tjenester [43]. RADIUS en nettverksprotokoll som kan sentralt håndtere autentiserings-, autorisasjons- og regnskapsmanagement for brukere som ønsker tilgang til nettverkstjenester.



Figur 10: Scenario: Tilgang til kjernetjenester for TKN.

Hvordan SDNi funksjonaliteten vil fungere og eventuelt kunne håndtere forsinkelser på grunn av smalbåndslinjer eller radiokommunikasjon er usikkert. Kontroller ved MTKN kommuniserer med kontroller i kjernen ved hjelp av SDNi eller annen proprietær SDN protokoll, en proaktiv tilnærming for distribusjon av dataflyt medfører til at MTKN kan fungere autonomt og kun endringer i topologien formidles til og fra kjernen. Tilgang til tjenester av prinsippet «ad-hoc» er mulig å få til: Kontroller ved MTKN kommuniserer med kontroller i kjernen ved hjelp av SDNi der distribusjon av dataflyt blir iverksatt hvis bruker har nødvendig autorisasjon. Figur 11 illustrerer kommunikasjonen mellom MTKN og kontroller i FKI. Figur 11 beskriver kun den virtuelle kommunikasjonen:



Figur 11: MTKN virtuelle kommunikasjon for tilgang til taletjenester.

1) MTKN sender forespørsel til kontrolleren i kjernen.
2) Det settes opp en sikker forbindelse mellom kontroller i kjernen og noden, bruker må autentiseres.
3) Brukerens autorisasjons parametere (brukernavn, passord, type tjeneste m.m.) blir videresendt av kontroller til RADIUS der autentisering og autorisasjonslogikken befinner seg. RADIUS kan velge å akseptere eller avslå tilgang. Akseptere den tilgang blir autorisasjonen sendt til kontroller sammen med informasjon om start av regnskap.
4) Kontroller implementerer autorisasjonen, setter opp dataflyt og formidler dette til kontroller ved MTKN sammen med autorisasjonen.
5) MTKN kan deretter kommunisere direkte med tjeneren der taletjenestene befinner seg. Brukerens bruk av tjenester blir regnskapsført basert på informasjonen som blir logget fra sesjonen som er opprettet mellom RADIUS og kontrollerne. Når MTKN avslutter sender MTKN kontrolleren «status_type=stop»[49]til kontroller i kjernen som videresender den til RADIUS som avslutter regnskapet og avslutter sesjonen.

Kommunikasjon mellom kontroller og RADIUS tjener må ha integritet- og konfidensialitets beskyttelse. Autorisasjonsparametere blir overført her, forbindelsen må derfor være kryptert.

Logikken for funksjonen hos kontrolleren i FKI kan beskrives som følgende:

Funksjonalitet	**Logikk / funksjon / algoritme**
Handshake mellom noden og kontroller i kjernen.	Sjekk av sikkerhetsparametere.
Bruker autorisasjon.	Oppgi brukernavn og passord.
Brukernavn og passord mottatt, sjekker autorisasjon med RADIUS. Kontroller sender tilbakemelding til bruker, godtar eller avslår, avslutter forbindelsen.	Forespørsel (Fra kontroller). Tilgang gis/ikke, autorisasjonsparametere (policy, bruker-tjenester).
Tilgang og autorisasjon gis noden.	Dataflyt distribueres til kontroller på TKN.
Regnskapsføring av aktiviteten til bruker starter. Bruker avslutter.	Send start regnskap til RADIUS. Sender stop når bruker er ferdig å bruke tjenestene. Forbindelsen termineres.

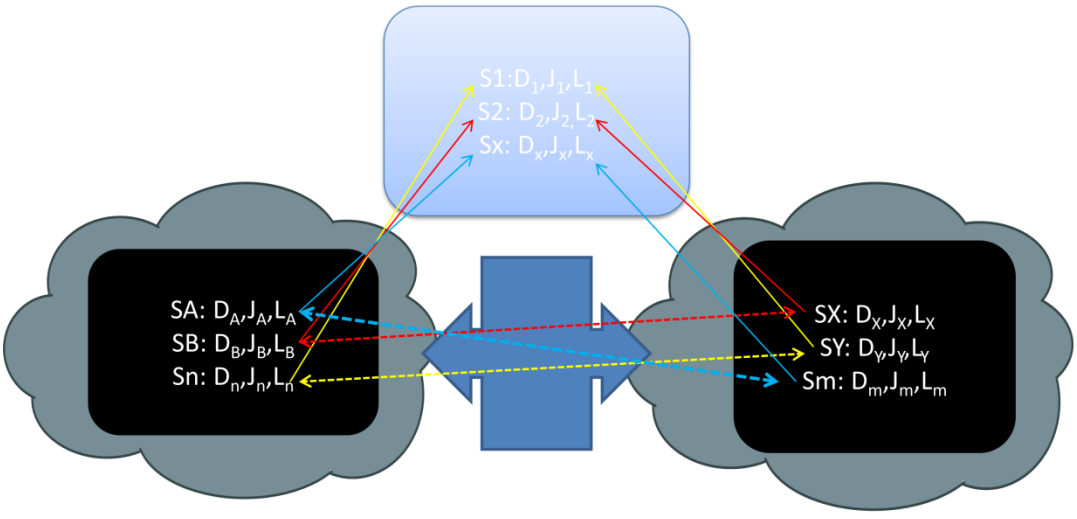
Tabell 3: Kontroller logikk

Kontroller i kjernen håndterer kommunikasjonen og fungerer i dette designet som *gatekeeper* mellom MTKN, RADIUS og kjernetjenestene. API funksjonalitet mot bruker må utvikles og tilpasses kontrolleren som velges.
AAA-tjeneste logikken kan også distribueres på noder som har egne kontrollere. En installasjon av RADIUS på MTKN kan autentisere noder i TOS som ønsker å benytte tjenester i MTKN.

5.5 Militær prioritet for TKN

For å kunne håndtere differensiert tjenestekvalitet på tjenestene som har behov for militær prioritet krever det en detaljert bevissthet over trafikken i nettverket (ref. Kapittel 2.3.2). Målsetningen med å ha en definert QoS policy er å oppnå ende til ende QoS i nettverket ved at trafikken får lik eller tilsvarende lik behandling gjennom hele nettverket.

NATO[12] Standardization Agency (NSA) har utgitt en Interperability Point and Quality Of Service Standardization Agreement (IOP QoS STANAG) [11] og en teknisk rapport [13] for Federation of Networks (FoN), hvor målsetningen er at flere nasjoner med individuell nettverksstruktur (taktiske og strategiske nettverk) skal kunne kommunisere med hverandre med ende til ende tjenestekvalitet på tjenestene. STANAG-en heter STANAG 4711 og rapporten TN 1417 og beskriver overordnet en helhetlig policy for QoS som kan implementeres på tilknytningspunktene mellom nettverkene. Denne policyen kan også benyttes på nasjonale grensesnitt slik at datapakker fra samme type tjeneste får lik merking (figur 12), dette muliggjør militær prioritet på tjenester ved eventuell metning eller kø i nettverket.

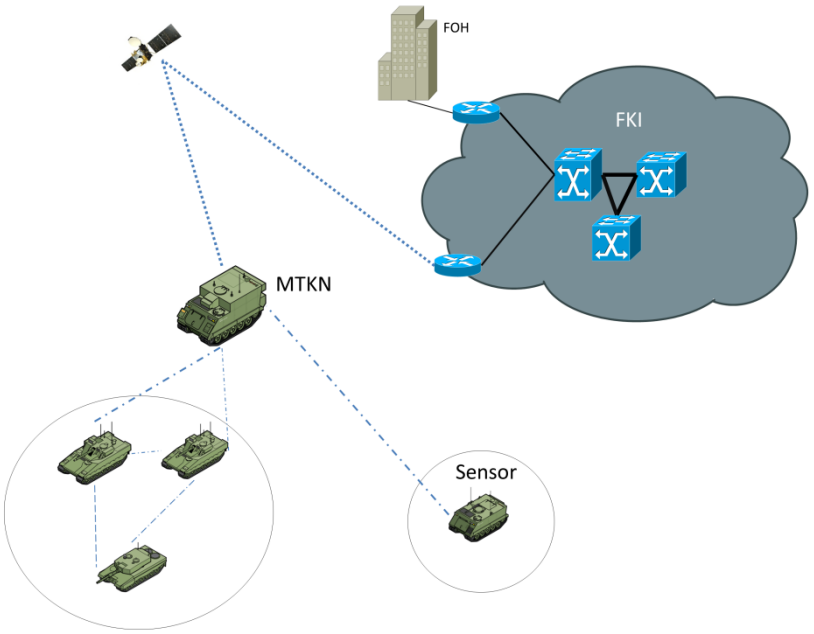


Figur 12: IOP mapping.

Under følger et scenario og løsningsdesign med bruk av SDN.

[12] The North Atlantic Treaty Organization (NATO)

5.5.1 Scenario: Differensiert tjenestekvalitet for TKN



Figur 13: Scenario: Differensiert tjenestekvalitet for TKN.

Figur 13 illustrerer et kjøretøy i et operasjonsområde som fungerer som MTKN, den er også relé for andre kjøretøy. Den mobile enheten kan ha *ultra high frequency* (UHF) radiokommunikasjon med de andre mobile enhetene og er forankret til FKI ved hjelp av en SATCOM forbindelse eventuelt *high frequency* (HF) radioforbindelse. Et kjøretøy fungerer som sensor og har mulighet til å sende videostrøm. Forsvarets operative hovedkvarter (FOH) ønsker å få sanntidsinformasjon fra sensor-kjøretøyet og det har militær prioritet.

Utfordringen i nettverket i figur 13 oppstår når det dukker opp behov under en operasjon som ikke er beskrevet i CONOPS og konfigurasjonen implementert er ikke klargjort for denne tjenesten.

Scenarioet over er komplisert siden nettverket kan ha flere topologier og kapasitetene mellom noder og MTKN og FKI er varierende. Med bredbånds UHF radio vil sensorkjøretøyet ha nok av kapasitet til å sende videostrømmen til MTKN men kapasiteten reduseres betraktelig der. MTKN har flere tjenester som er tilknyttet kjernetjenestene i FKI, i tillegg mottar den data fra alle kjøretøyene som kan benytte samme bærer. I denne dynamiske infrastrukturen og med eventuelt MANET topologi er det vanskelig å kunne gi militær prioritet på videotjenesten (ref. Kapittel 2.3.2).
QoS implementasjon for MPLS baserte trådløse nettverk der MPLS befinner seg i kjernen kan gi QoS og *Class of Service* (CoS) funksjoner:
«(...) MPLS QoS architecture can be employed to provide traffic engineering in broadband

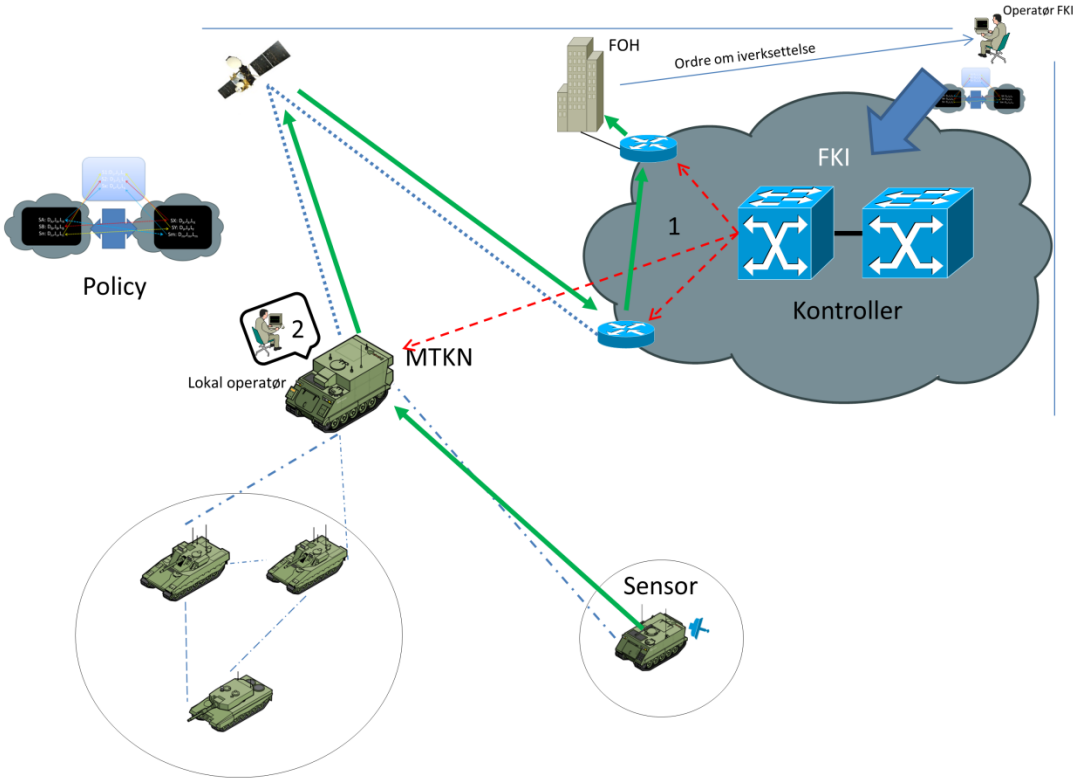
wireless networks (...)»[50]. En forutsetning for å få dette til er bredbåndskapasiteter på det trådløse nettverket, årsaken er mengden av management trafikk mellom kjernenettverket og nodene.
I dette scenarioet (ref. Figur 13) kan kapasiteten fra mobile noder til MTKN være hundre ganger høyere enn kapasiteten mellom IOP og kjernenettverket. Management trafikk mellom noder og fra noder inn til kjernen må derfor holdes til et minimum.
Metoden for å understøtte det tjenstlige behovet er å sørge for å ha stor nok båndbredde kapasitet, ref. Kapittel 2.3.1: *«Bygg ut kapasitet i takt med behovet».* Hvis kapasiteten er underdimensjonert vil FOH i dette tilfelle ikke kunne motta videostrømmen.

5.5.2 Design med av SDN

MTKN fungerer som IOP og den bør ha implementert en policy som beskriver tjenestekvalitetene på tjenestene. IOP skal kunne se når trafikk med militær prioritet ankommer noden og handle deretter (ref. Kapittel 2.3).
Ved hjelp av SDN skal det være mulig å styre nettverket slik at policy for tjenestekvalitet kan distribueres og iverksettes til kontrollere eller rutere i nettverket (ved hybrid SDN nettverksløsning). Videre skal nødvendige dataflyt bli distribuert til svitsjer i nettverket slik at datapakker får korrekt behandling (nødvendig prioritet), og ende til ende QoS blir opprettholdt.

Flaskehalsen i nettverket er en vesentlig faktor som må SDN designet må ta hensyn til. Management trafikk må holdes til et minimum.

Ved hjelp av SDN kan dette løses på følgende måte, figur 14:



Figur 14: Etablering av policy for differensiert tjenestekvalitet med SDN.

Alternativ **1**: Fra et management grensesnitt (API) kan det iverksettes en policy som kontroller i kjernenettverket i FKI som har komplett topologioversikt, distribuerer til kontrollerne i hierarkiet som aktivisere den slik at datapakken får samme behandling gjennom hele nettverket, kontrolleren setter også opp eventuelle dataflyt. Policy må kunne iverksettes av en nettverksoperatør som sitter med det komplette situasjonsbildet av nettverket i FKI.

Alternativ **2**: Bruker iverksetter policy fra en applikasjon på en management klient på IOP noden. Kontroller ved IOP kommuniserer deretter med kontroller i kjernen for iverksettelse av policy.

1. For begge alternativene:

   – Kontroller mottar kall fra nordgående grensesnitt. Setter opp sikker forbindelse.

   ➢ Autentisering og autorisasjon sjekkes, gir bruker korrekt tilgang.

- Kontroller kommuniserer til nettverksinfrastrukturen på sørgående grensesnitt ved hjelp av en management protokoll (OpenFlow, onePK etc).

- Kontroller får kontakt med korrekt grensnitt og setter opp sikker forbindelse.

- Nye dataflyter blir satt opp i tabellen ved nettverksenheten og policy blir iverksatt på kontroller.

2. Når denne rapporten ble skrevet støttet SDN kun *DiffServ* klasse basert trafikk (ref. Kapittel 3.1.4 og 4.1.4).
   Når datapakker ankommer IOP vil pakkene bli oversatt *(mapping)* til korrekte verdier (Type Of Service, TOS) fra kontroller, lagt i korrekt kø og sendt ut på grensesnittet basert på prioriteringen av datapakken. Målsetningen er å oppnå følgende sluttilstand ref. kapittel 2.3.2:
   «Når nettverket har stor belastning må IOP ha mulighet til å kunne endre policy eller iverksette en «minimum mode» slik at viktig trafikk, militær trafikk, med høy prioritet fortsatt får tjenestekvalitet i motsetning til mindre viktig trafikk.»

Scenario 2, merking av trafikk metoden kan beskrives slik i pseudo kode (tabell 4):

```
Program StartPolicy QoS:
(..)

Lag policy()
   Policy ingress
     Ingress tabell
        Match:
            function match CS()
            function match DSCP()
        Action:
            Merk CS til DSCP()
            Merk DCSP til DCSP()
     Policy egress
        Egress tabell
           Match:
               function match DSCP()
           Action:
                If (Ingen match)
                     drop pakker
                else
                  function Klassifiser kø (DSCP)
                     Sett pakker i riktig kø(båndbredde, terskel,kø størrelse)
```

Tabell 4: QoS policy i pseudo kode.

En forutsetning for at policyen skal kunne fungere må det eksisterer et konsept for merking av trafikk, først og fremst i brukersystemene i de taktiske nettverkene hvor sannsynligheten for metning og forsinkelser er større enn i transport- og kjernenettverket. Merkingen må skje fra applikasjonene i brukersystemene eller på

portene hvor de er tilknyttet. Trafikk som beveger seg mellom IOP og nodene, mellom taktiske nettverk eller taktiske og strategiske nettverk, får en uniform merking ved hjelp av ST 4711 policyen.
Trafikk som blir ikke merket blir behandlet som *best effort*, militær prioritet vil dermed ikke kunne oppnås.

5.6 Oppsummering

Dette kapittelet har beskrevet foreslåtte SDN design på to scenarioer for Taktiske kommunikasjonsnoder med forankring til FKI.
Neste kapittel beskriver evalueringsmetodene for SDN og valgt metode for denne oppgaven.

6. Evaluerings metoder

I dette kapittelet vil jeg se nærmere på evalueringsmetodene som kan benyttes for SDN. Først ser jeg overordnet på metodene deretter diskuterer jeg metoden jeg har valgt for evaluere SDN for denne oppgaven.

6.1 Introduksjon

Ved utvikling av nettverksteknologier og kommunikasjonsprotokoller for SDN består mye av forskningen på evaluering og analyse av designet, og vurdere om ny tilnærming er bedre eller dårligere enn eksisterende sammenlignbare løsninger.
Fire velkjente teknikker eksisterer for å hjelpe og evaluere og analysere nettverksteknologier og protokoller:

- Simulering
- Analytiske modeller
- Emulering
- Praktiske eksperiment

Hvert av punktene over har sine respektive fordeler og ulemper. De blir kort diskutert under.

6.2 Simulering

Simulering er en godt etablert og mye benyttet metode for å gjennomføre evaluering av ytelse og robusthet av nettverkskomponenter og protokoller. GNS3 [51], PacketTracer [52] og NS3 [53] er noen eksempler på simuleringsverktøy som har støtte for de mest populære nettverksprotokollene eksempelvis IP, TCP, UDP, Ethernet etc. Flere protokoller i protokoll modellene kan bli simulert samtidig og gir muligheter for å teste implementasjoner av egne scripts, koder, algoritmer eller test av nye protokoller. Simulering gir mulighet til å kjøre tester flere ganger for å verifisere resultatene, samtidig er den skalerbar og kan simulere mange noder i nettverket. Begrensninger ligger som regel i ytelsen på plattformene som simuleringene blir kjørt på.
Simuleringsmodeller vil simplifisere en del nettverksoppsett og den fysiske infrastrukturen da det er vanskelig og simulere alle effekter som du vil ha i et operasjonelt nettverk, spesielt i trådløse og taktiske nettverk. Anslag og antakelser blir gjort og konklusjonene av resultatene kan bli forutinntatte.

Nettverksprotokollene ville også være forenklede utgaver av den originale. Simuleringer kan benyttes når det ønskelig å teste en spesifikk funksjonalitet, protokoll eller algoritme.
For simulering av nettverk til bruk for enkelte versjoner av SDN kan man bruke GNS3 eller NS3. Noe å bemerke seg, NS3 støtter kun versjon 1.0 av OpenFlow [54].

6.3 Analytiske modeller

Analytiske modeller blir ofte benyttet for å evaluere spesielle egenskaper av protokoller, algoritmer eller beskrive trafikk mønster. Analytiske modeller er utbredt da de er presise, ressurs effektive og ofte gjenbrukbare i problemstillinger som er ganske like. Derimot kan modellene være vanskelig å forstå uten å ha dybde kunnskaper i fagområdet.
Analytiske modeller er blant annet brukt for å gjennomføre ytelses målinger av kontrollere i SDN baserte nettverk [55, 56].

6.4 Emulering

Ved bruk av emulering blir maskin- og programvarekomponenter opprinnelig designet for distribusjon i operasjonelle nettverksmiljøer kombinert med simuleringskomponenter. Emulering av nettverk er et viktig aspekt for forskning og utvikling i fagområdet nettverk og kommunikasjonssystemer. Det er klare fordeler å bruke en emulator fremfor en simulator. En simulator vil prøve å gjenskape for eksempel en ruter ved hjelp av et programmeringsspråk, mens en emulator tar det originale operativsystemet til ruteren som blir implementert og emulerer maskinvaren under den, på den måten kan en ruter bli kjørt på en ordinær datamaskin.
Målsetningen er å få testet protokoller og algoritmer på maskinvare plattformer før eventuelle praktiske eksperiment. En emulator kan fungere på flere lag i protokoll-modellene og parameterne kan endres underveis. Emulering kan bli gjort i ulike testoppsett for det spesifikke som ønskes å forskes på og kan være referanseanlegg for en skarp implementasjon. Emulering er derfor et verdifullt kompromiss mellom simulering og praktiske eksperiment med tanke på kostnader, tid og pålitelighet. Et eksempel er kode på et emulert nettverk i et testoppsett som kan omsettes raskt i operasjonelle nettverk.
Det finnes flere type nettverks- simulatorer og emulatorer for SDN. Hvilken som velges må sees i sammenheng med valget av SDN kontrolleren og størrelsen på det simulerte nettverket. EstiNet og Mininet er to eksempler som er ganske utbredt for SDN OpenFlow type kontroller.

6.5 Praktiske eksperiment

For å kunne validere analytiske modeller, algoritmer og simuleringer er praktiske eksperiment en nødvendighet, spesielt skal det være mulig evaluere alle aspekter av nettverksdesignet. Men praktiske eksperiment er ressurs intensiv, tidkrevende og kostnadsmessig mye høyere enn de andre evalueringsmetodene. Resultatene fra eksperimentene er ofte vanskelige å validere på grunn av alle faktorene som må tas hensyn til. Praktiske eksperiment krever god planlegging i forkant og testplaner bør spesifiserer hva som skal testes og valideres i nettverksoppsettet.

6.6 Kildemateriale

Litteraturstudie som er gjort i denne oppgaven kan deles opp i fire hoveddeler. For å kunne forstå Forsvaret sin nettverksinfrastruktur var det nødvendig å se nærmere på Forsvarets IKT-policy og IKT-strategi. De gir grunnleggende forståelse for hvordan IKT i Forsvaret skal kunne benyttes til å understøtte militære operasjoner og konsepter. Deretter ble Forsvaret sin egen kommunikasjonsinfrastruktur og prosjektet for Taktiske områdesamband studert nærmere. De gir en oppfatning av hvilke teknologier Forsvaret benytter i sine nettverk i dag, hvilke utfordringer de har i sin infrastruktur og hvilke planer de har gjennom noen materiellprosjekter. NATOs arbeid innenfor det taktiske domenet ble også sett nærmere på. Tilslutt startet forskningen av SDN og en gjennomgang av en rekke publikasjoner og noen lærebøker. Under mitt arbeid med oppgaven eksisterte det ikke så mange lærebøker om SDN. Jeg vurderer SDN som et ganske nytt konsept innen fagområdet nettverksteknologier. Alle delene av litteraturstudie har gitt et viktig grunnlag for forskning og utvikling av denne avhandlingen.

6.7 Evalueringsmetoder for SDN

Evaluering av ulike SDN design kan bli utført ved hjelp av alle evalueringsmetodene beskrevet over, antatt at utvikleren har tatt i betraktning alle fordeler og ulemper. Gjennom litteraturstudiet av denne oppgaven er emulering og simulering de mest utbredte metodene for å evaluere SDN konsepter og design. Men studiene er vanskelige å sammenligne ettersom de tester spesifikke elementer, applikasjoner eller funksjonalitet i SDN og er tilpasset egne scenarioer. Isolert sett, en evalueringsmetode er ikke gyldig for alle typer løsninger. Hensynet til kostnader, ressurser og tidsforbruk som øker ved gjennomføring av emulering fremfor simulering og blir vesentlig større hvis det besluttes å gjennomføre praktiske eksperiment, vil også påvirke valget av metode.
SDN kontrollerne er programvarebaserte, det betyr at de har forskjelligartet støtte for programmeringsspråk. Tilgjengelig dokumentasjon er også forskjellig fra leverandør til leverandør. Disse momentene blir også viktige.

For å kunne reflektere bedre over valget av metode bør noen faktorer for SDN vurderes. Tabell 5 gir en oversikt:

Faktor	Beskrivelse
Hvilken kommunikasjonsprotokoll skal SDN løsningen ha støtte for: OF, onePK, NetConf eller flere?	OF standarden er en åpen standard og den mest utbredte av standarder på SDN nettverksutstyr. Leverandør utviklete SDN kontrollere har også begynt å støtte OF protokollen. Men hvilken versjon av OF skal støttes? Ulike versjoner av OF gir forskjellige funksjonaliteter. Hvilken leverandør har støtte for protokollene og hvilke versjoner? Hvilke egenskaper ønskes til egen nettverksløsning?
Valg av leverandør	Hvor sterk aktør i markedet, ambisjoner. Hva er leverandørenes plan for implementasjoner?
Hvor skalerbar bør løsningen være?	Antall svitsjer som SDN kontrolleren bør ha støtte for. Er nettverket dynamisk?
Hva med nettverksfunksjonaliteten?	Hvordan er nettverkstopologien, skal det implementeres tjenestekvalitet (QoS) parametere. Hvilken tilnærming for distribusjon av dataflyt er mest hensiktsmessig for denne løsningen?
Skal løsningen være programmerbar?	Skal det være mulige å kunne ta i bruk maler, scripts og egne utviklete APIs.
Ønskes det støtte for virtualisering?	I forhold til skalering så gir virtuelle nettverkstopologier muligheter for dekobling fra den fysiske nettverkstopologien.
Pålitelighet	Antall kontrollere i nettverket og funksjonene den skal ha. Hva med oppsettet av topologi og eventuelle cluster løsninger.
Sikkerhet	Plassering av kontroller i arkitekturen, antall kontrollere. Skal løsningen kunne autentisere og gi nødvendig autorisasjon av brukere til nettverket.
Sentralisert monitorering	Skal løsningen monitorere kun deler av nettverket eller hele?
Hva med ytelsen?	Ytelsen på kontroller for håndtering og beregning av dataflyt. Ytelse på grensesnittene (I/O) og ved oppsett av forbindelser i forhold til forsinkelser.

Tabell 5: Faktorer for SDN.

6.8 Valgt metode

Noen av faktorene påvirker valget av evalueringsmetode for denne oppgaven. Dette har sammenheng med at en del av funksjonaliteten til SDN kontrolleren blir bestemt fra faktorene som blir vurdert som viktig for nettverket.
I denne oppgaven blir en kombinert tilnærming ved hjelp av emulering og simulering benyttet. Ved hjelp av emulering er det mulig å etablere et testoppsett som kan benyttes som et referanseanlegg for eventuelt praktisk implementasjon. Det legges vekt på at metoden må kunne lage virtuelle nettverk som er lik eller tilnærmet lik scenarioene beskrevet i kapittel 5.
Testoppsett med fysisk nettverksutstyr og nødvendig programvare er en tidkrevende prosess og kan generer en del kostnader, langt mer enn ved emulering. Prosessen underveis krever ikke minst god konfigurasjonskontroll siden det er vanskeligere å *nullstille* et praktisk oppsett hvis det oppdages feil i programmeringskoder, ved systemkrasj eller når feilsøking ikke klarer å løse problemet innenfor nødvendige tidsrammer.

6.9 Valgt design

Det må gjennomføres en undersøkelse av SDN kontrollere som tilfredsstiller kravene til implementasjonen som er beskrevet ved hjelp av scenarioene i kapittel 5, faktorene for SDN og valgt metode.
Videre må et testoppsett etableres for implementasjon av SDN.

Applikasjon og funksjonalitet for AAA-tjenester må utvikles. Applikasjonen må kunne håndtere brukerautentisering og autorisasjon ved å regulere tilganger til tjenester og starte og stoppe regnskapstjenesten. Applikasjonen må kunne kommunisere mellom bruker og kontroller, og mellom kontroller og RADIUS.

Det må også utvikles en applikasjon og funksjonalitet for distribusjon av QoS policy. Applikasjonen bør fra et brukergrensesnitt kunne iverksette policy på et grensesnitt i nettverket. Policy må kunne merke trafikk og gi nødvendig differensiert tjenestekvalitet og militær prioritet.

Det må etableres et nettverksmiljø som kan benyttes til å validere funksjonaliteten av implementasjonen. Nettverket kan etableres ved hjelp av nettverksemulator.

6.10 Oppsummering av evaluering og valg av metode

Dette kapittelet har vært en overordnet gjennomgang av evalueringsmetodene som kan gjennomføres for denne oppgaven. Jeg har også beskrevet noen av faktorene som må vurderes for SDN ved valg av metode og funksjonalitet til kontrolleren. Tilslutt ble metoden valgt for denne oppgaven.

7. Implementasjon

Dette kapittelet er en beskrivelse av min implementasjon av SDN løsningen.
Først ser jeg nærmere på noen SDN kontrollere og gir en kort vurdering av dem. Videre beskrives SDN kontrolleren som jeg valgte, hvilket verktøy jeg brukte for å lage testoppsettet, deretter kommer funksjonaliteten jeg utviklet for scenarioene (ref. Kapittel 5). Tilslutt kommer litt om nettverkstopologien jeg testet min implementasjon på. Testene som ble gjort, valideringen og resultatene av testene blir beskrevet nærmere i kapittel 8.

7.1 SDN kontrollere

Ved valg av SDN kontroller til mitt testoppsett ble flere momenter vurdert: Forskningen, SDN arkitekturen i FKI og scenarioene for TKN og metoden (ref. Kapittel 4, 5 og 6). Spesielt viktig var også tilgjengelig dokumentasjon hos leverandør, programmeringsspråk kontrolleren støttet, standardiseringer på eventuelt nordgående og sørgående grensesnitt og mulighetene for distribuert arkitektur. I tillegg ble utvikler forum eller support hos leverandøren og pågående utvikling tatt hensyn til.
I tabell 6 har jeg sett på noen vanlige kontrollere og programmeringsspråket de støtter.

Navn på kontroller	Beskrivelse
Beacon [57]	Beacon er en OpenFlow kontroller implementasjon skrevet i Java og er integrert i OSGi [58] rammeverket. Dette gir utviklere mulighet til å legge til, bytte ut eller fjerne kode selv når kontrolleren kjører. Dokumentasjon er mangelfull, den har heller ikke hatt noen stor utvikling de siste årene.
NOX [59]	Var den første referanse implementasjonen av en OpenFlow kontroller. Den ble skrevet i Python og C++, eksisterer i dag kun i C++.
POX [60]	Er en evolusjon av Python OpenFlow grensesnittet som ble fjernet fra NOX. Kan kjøre på flere ulike plattformer. Har hatt liten utvikling de siste årene.

Floodlight [61]	OpenFlow kontroller som er skrevet i Java, ble avledet fra Beacon prosjektet og har en forenkling av OSGi rammeverket slik at den skal være enklere å benytte for utviklere.
OpenDaylight [62]	Er en OpenFlow kontroller som er skrevet i Java på OSGi rammeverket [63]. Er et *open-source* samarbeidsprosjekt med The Linux Foundation, er en stor aktør på markedet.
Cisco One Platform Kit (onePK)[64]	Dette er Cisco sitt første element i deres Open Network Environment SDN strategi. onePK er et verktøy for programutviklere som gjør det mulig å få tilgang til, utvide eller tilpasse programvare funksjonaliteten levert av Cisco-rutere og svitsjer. Den har bibliotek for C, Java og Python. onePK har støtte for OpenFlow agenter i nettverket.
Trema[65]	Er et rammeverk for utvikling av OpenFlow kontrollere, skrevet i Ruby og C.

Tabell 6: SDN kontrollere

Det var to kontrollere som skilte seg ut i forhold til de andre. Både OpenDaylight og Cisco onePK hadde mye dokumentasjon, utvikler forum og støtte for mye funksjonalitet. Begge hadde også støtte for JAVA programmeringsspråk som jeg foretrekker å bruke. På grunn av standardiseringene som var gjort i arkitekturen på kontrolleren Cisco onePK, vurderte jeg onePK som den beste kandidaten for min oppgave.

7.2 onePK SDN kontroller

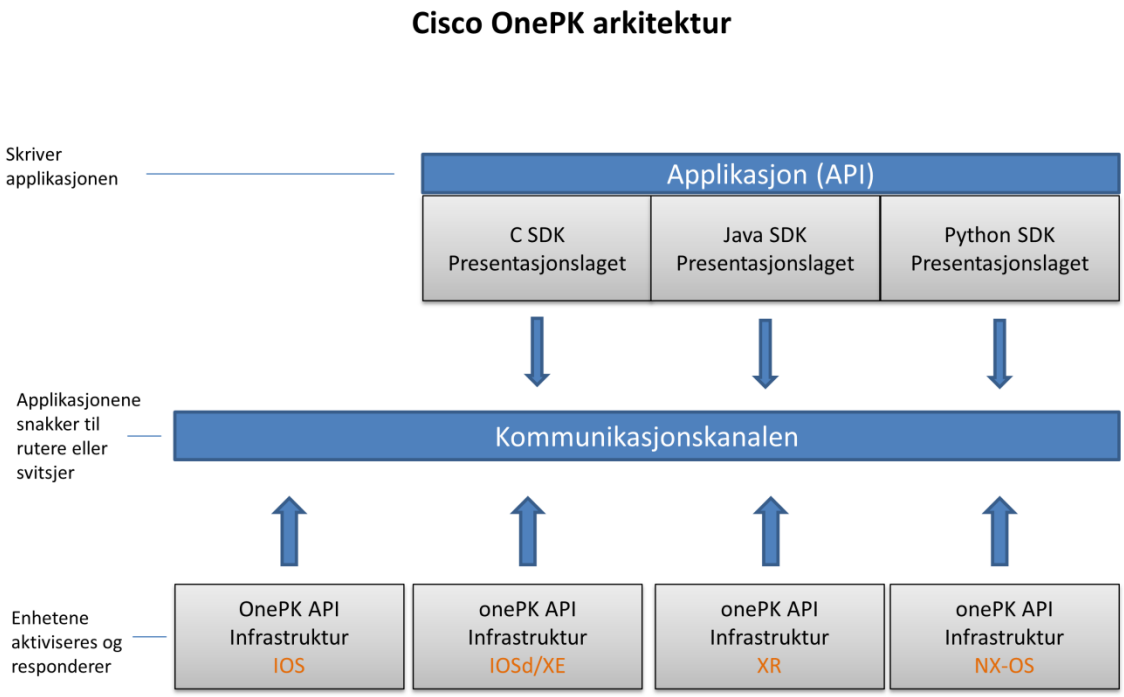
Cisco sin SDN arkitektur er kalt Cisco *Open Network Environment* (ONE), der kontroller intelligensen i Cisco ONE ligger i maskinvaren mens nettverksadministrasjon ligger i programvaren.
One Platform Kit (onePK) er et utviklingsverktøy for programutviklere som ønsker å utvikle programvare for automatisering, vedlikehold, lage ny eller tilpasse funksjonalitet i egen nettverksinfrastruktur bestående av Cisco rutere og svitsjer. I prinsippet består det av et *Software Development Kit* (SDK) for Java, C og Python for utvikling av applikasjoner og tjenester på sitt eget nettverk. Konseptet for Cisco er å ha ett API sett for alle store plattformer: IOS, IOS-XR, IOS-XE og NX-OS [66].

Mens OpenFlow har begrenset funksjonalitet, kan onePK støtte et stort antall APIer som kan bruke intelligensen innebygget i Cisco sin maskinvare. Cisco støtter også OpenFlow standarden i onePK.

Programutvikleren har stor frihet til å velge selv utviklings- og kompileringsverktøy. Cisco har et forum der utviklere kan kommunisere med hverandre eller rette spørsmål direkte til Cisco sitt utviklingsteam. Kravet er at den enkelte må registrere seg på Cisco Development Network (DevNet).

7.2.1 onePK arkitektur

Den tekniske arkitekturen av onePK ser du av figur 15 under:



Figur 15: onePK arkitektur [66].

Arkitekturen består av tre hovedelementer:

1. **Presentasjonslaget.**

   Består av API biblioteket som programutviklere kan bruke på sine applikasjoner. Biblioteket har få avhengigheter som gir programutviklere ganske stor fleksibilitet for hvordan de ønsker å presentere dataene sine som er kommunisert fra underliggende lag. Samtidig gir det mulighet å forenkle grensesnittene for brukerne på det de ønsker å gjøre med nettverkselementene.

2. **Kommunikasjonskanalen.**

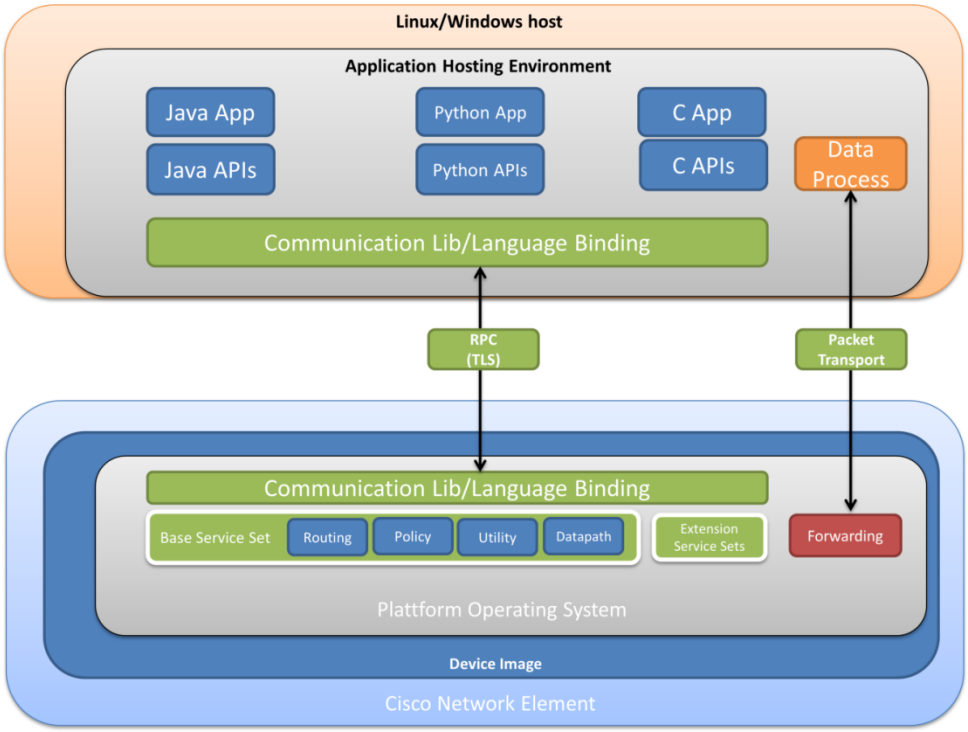
   Sørger for at kall fra presentasjonslaget blir kommunisert riktig og sikkert ned til infrastrukturen, tilsvarende sørger for at data fra underliggende lag blir kommunisert til presentasjonslaget.

3. onePK API infrastruktur.

API infrastrukturen får tilgang til funksjoner som er interne til en ruter eller en svitsj. API infrastrukturen sørger for at funksjonskall vil kunne fungere på tvers av Cisco plattformer: IOS, OIS XR, IOS XE eller NX-OS.

Systemarkitekturen gir en tjenesteimplementering som onePK klient-server modellen gjør tilgjengelig i en plattform- og enhetsuavhengig måte.
Applikasjonen (*Application Hosting Environment*) sitt tjenestebibliotek kan kun kommunisere med sin plattform spesifikke motpart på nettverksenheten ved hjelp av *Remote Procedure Calls* (RPC)[67]. Nettverksenheten samhandler deretter med tilsvarende kommunikasjonsbibliotek til en plattform spesifikk tjenesterammeverk som kommuniserer med maskinvaren til Cisco. Figur 16 gir en illustrasjon av dette og de andre systemkomponentene i onePK.

Applikasjoner som blir utviklet kan kommunisere med nettverksenhetene over en sikker tilkobling, eksempelvis TLS til en onePK tjener. onePK systemet er et klient-tjener system, hvor tjeneren kjører på Cisco nettverksenheter som en prosess på Cisco operativ systemet (OS). Klienten er applikasjonen eller applikasjonskoden fra onePK klient biblioteket.

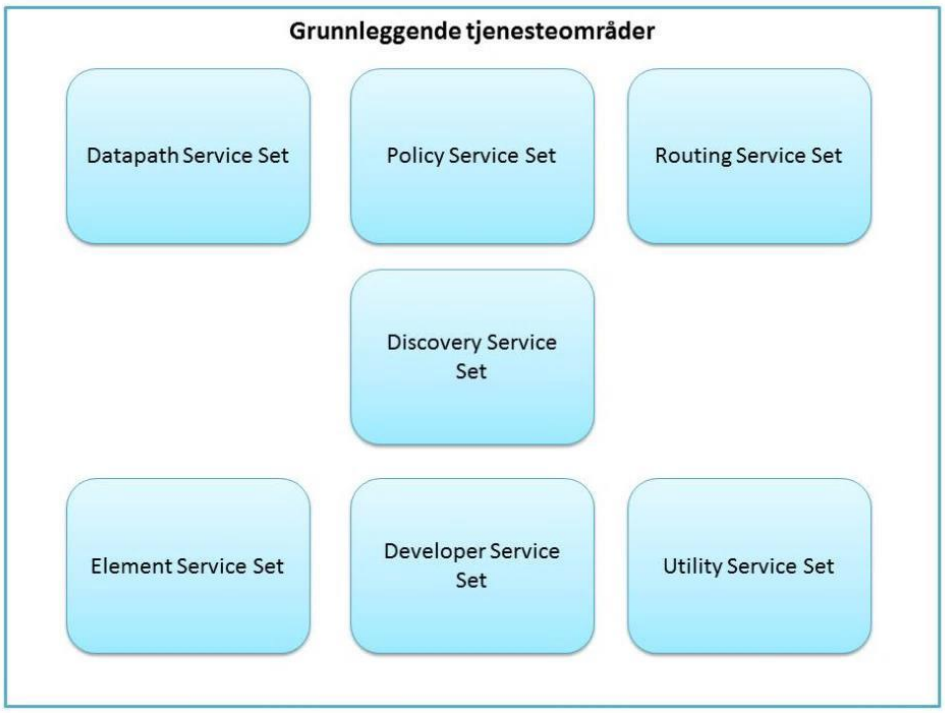


Figur 16: Systemkomponenter [63].

Tjenesteområder

onePK SDK for vekselsvis Java, C eller Python består av betydelig mange klasser og er veldig komplekst. Programmeringsferdigheter i tillegg til kompetanse på nettverk, arkitektur og eventuelt Cisco er etter min vurdering fordelaktig å inneha.
For å gjøre det lettere for utvikleren er funksjonaliteten til onePK organisert i tjenesteområder. Det er nettverkselementet som implementerer funksjonaliteten fra tjenesteområdene i henhold til kravene til maskinvarekombinasjonene.

Figur 17 er en illustrasjon av de grunnleggende tjenesteområdene. De er også kort beskrevet i tabell 7:



Figur 17: Grunnleggende tjenesteområder [26].

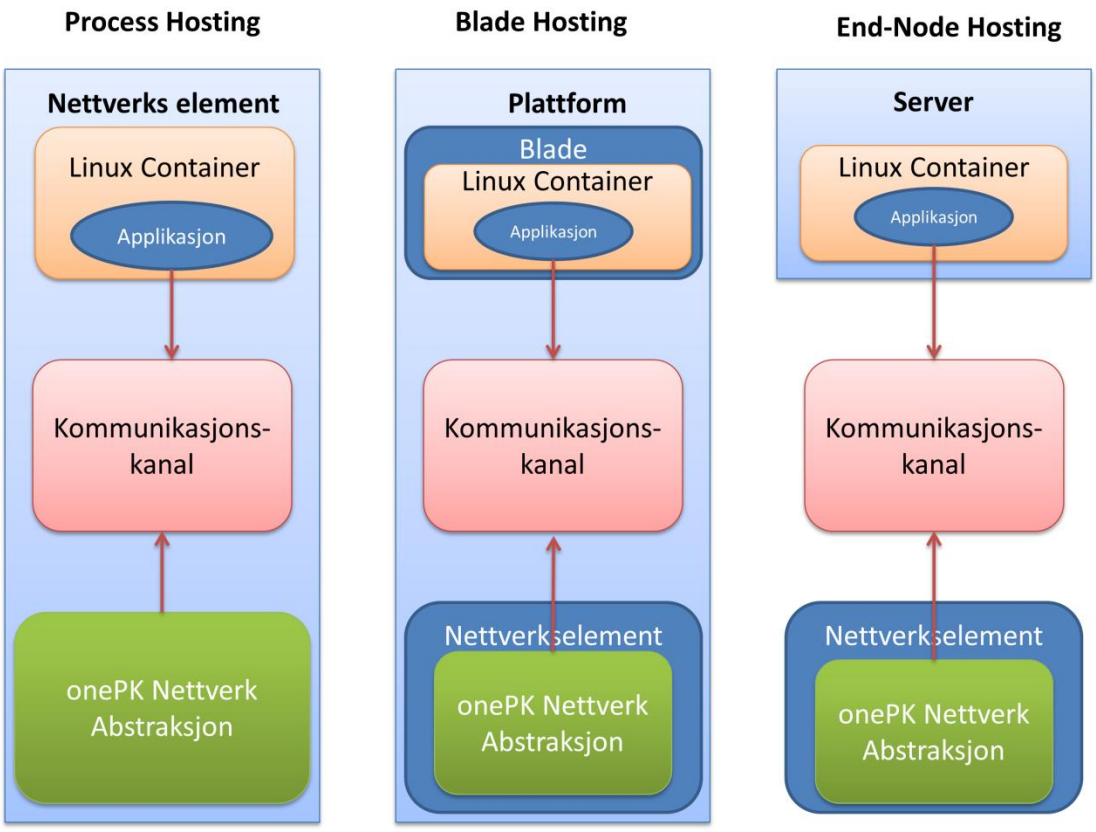
Tjenesteområde	Beskrivelse
Element Service Set	Har ansvaret for forbindelsen til nettverkselementene og gjør den tilgjengelig for alle andre tjenester og applikasjoner.
Discovery Service Set	Har mekanismer for nettverkstopologien og kan gjennomføre spørringer til nettverkselementet.
Policy Service Set	Har blant annet funksjonalitet for å legge til ACL eller QoS policy til et nettverkselement.
Routing Service Set	Leser *Routing Information Base* (RIB) informasjon, oppdatere ruting.
Datapath Service Set	Har funksjonalitet for pakke eller dataflyt klassifisering. Gir mulighet for en applikasjon å koble seg inn i pakkestrømmen på en Cisco svitsj eller ruter og ekstrakter datapakker. Disse pakkene kan enten kopieres, fjernes eller bli avledet fra data banen til applikasjonen.
Developer Service Set	Gir logging og feilsøking funksjonalitet på forbindelser til nettverkselementer.
Utility Service Set	Gir blant annet mulighet for å kjøre tilpassede kommandoer til kommandolinje (CLI) grensesnittet til nettverkselementet.

Tabell 7: Tjenesteområder.

Mer informasjon om alle tjenesteområdene finnes på Cisco sitt *Development Net* [68].

7.2.2 Distribusjonsmodeller

Det er tre måter en onePK applikasjon kan bli distribuert på, vist av figur 18 under:



Figur 18: Distribusjonsmodeller av onePK applikasjoner.

- **Process Hosting:** Applikasjonen kjøres i et Linux miljø på nettverkselementet. Isolasjon mellom onePK applikasjonen og nettverksprosessene er gitt av *Linux Containers* [69] eller på noen plattformer full KVM/QEMU [70].
- **Blade Hosting:** Applikasjonen kjøres på dedikert maskinvare adskilt fra nettverkselementet men på samme plattform.
- **End-Node Hosting:** Applikasjonen blir kjørt fra brukerens egen plattform, som kan være alt fra små mobile enheter til større server plattformer.

7.2.3 Oppsummering Cisco onePK

Stor fleksibilitet med egen programkode

Et element av Cisco Open Network Environment Software-Defined Networking Strategy er onePK. Ved hjelp av onePK kan programutviklere automatisere og lage egen funksjonalitet for sine Cisco baserte nettverk. Ved hjelp av et sett med APIer, som lar programmerere ta direkte funksjonskall i sin egen applikasjon, gir store muligheter for samhandling med nettverket. Programmere kan bytte ruter i nettverket, implementere policyer som avgjør hvem som har tilgang til hvilke opplysninger, håndheve regler for sikkerhet eller QoS-regler og påvirke pakkene som strømmer gjennom nettverket.

Kryssplattform støtte og standardisering i arkitekturen

onePK koden har kryssplattform støtte. Det betyr at en kode kan fungere på flere Cisco baserte plattformstyper. onePK kontrolleren støtter flere typer sørgående protokoller inkludert OpenFlow og kan kommuniserer med et stort antall fysiske og virtuelle nettverkselementer. Den inneholder også bestemmelser for horisontal kontroller til kontroller kommunikasjon og koordinering.

Arkitekturen i onePK er enkel og strømlinjeformet i forhold til andre SDN arkitekturer, dette gjør det enklere å utvikle funksjonalitet tilpasset brukerens behov eller profil.

Kompleksitet

Selv om arkitekturen er enkel er kompleksiteten veldig høy. Jeg anbefaler gode programmeringsferdigheter, forståelse av nettverksarkitektur og kompetanse innen for nettverksteknologier tilsvarende Cisco CCNA sertifisering [13] som nødvendig.

Ved eventuelle implementasjoner av teknologien i Forsvaret, er min vurdering at Forsvaret vil ha behov for Development and Operation (DevOPS)[14] operatører som har kompetanse innenfor fagområdene: Programutvikling, nettverk (Cisco CCNA sertifisert) og IT operasjoner.

I neste kapittel beskrives testoppsettet og min programkode og funksjonalitet i onePK for scenarioene.

[13] Cisco Certified Network Associate (CCNA) sertifisering er en sertifisering som viser at du behersker alle grunnleggende teknologier for å drifte et Cisco nettverk.
[14] DevOps: Programvare utviklingsmetode med fokus på blant annet kommunikasjon, integrasjon, automatisering og informasjonsdeling.

7.3 Mulige emuleringsverktøy

Jeg ønsket et verktøy som kunne emulere nettverkselementer og plattformer så likt som mulig et operasjonelt nettverk med fysiske komponenter. Jeg måtte derfor vurdere hvilke emulering- og simuleringsverktøy som kunne lage et testoppsett bestående Cisco onePK SDN kontroller, nettverk og muligheter til å teste mine applikasjoner og funksjonaliteter.

GNS3 eller Mininet emulatorer som er tilpasset OpenFlow baserte kontrollere kunne ikke benyttes til denne oppgaven. Cisco har gjennom sitt Networking Academy utviklet Packet Tracer [71] som benyttes for å simulere nettverk. Dette programmet benyttes primært til å lære seg Cisco nettverk og var heller ikke egnet.
Jeg valgte tilslutt å benytte *NDE*er (Network Device Emulator) som er selvstendige virtuelle maskiner (VM) som jeg kan installere virtuelle IOS (vIOS)[15] på. Topologiverktøyet er levert av Cisco og heter vmcloud [72] og kan kjøres på en Ubuntu plattform. Vmcloud er tilgjengelig for registrerte brukere fra Cisco DevNet. Ved hjelp av «XML scripting» kunne nettverkstopologier etableres.

For å bygge mitt testoppsett valgte jeg å benytte VMware virtualiseringsprogramvare på en Windows plattform. VMware kunne kjøre mine VMer, deretter kunne jeg konfiguere nettverket. Ved hjelp av VMware er det enklere nullstille oppsettet ved systemfeil eller krasj. Det gir meg også mer fleksibilitet siden VMer kan kopieres og benyttes på flere ulike plattformer som har VMware programvare.

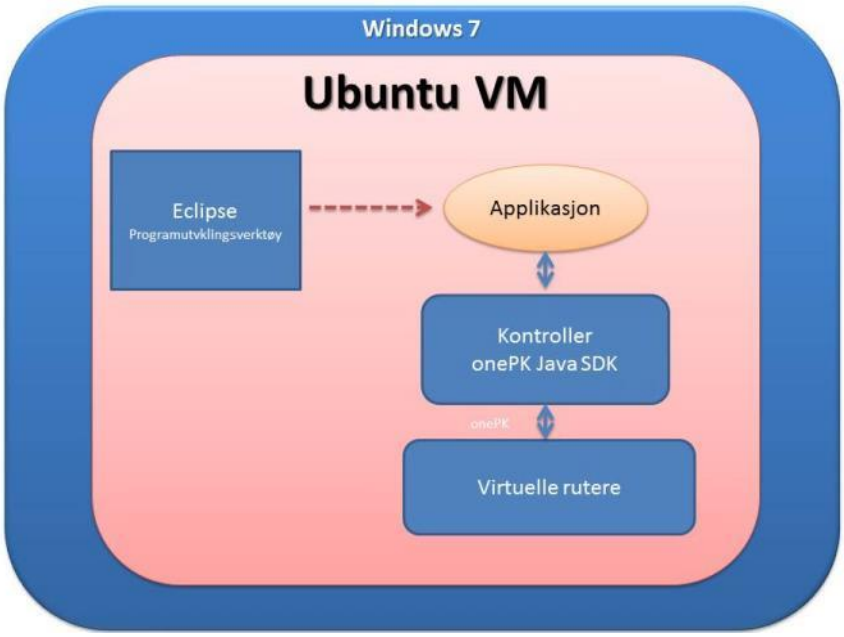
[15] IOS er et operativsystem som er installert på Cisco rutere og svitsjer.

7.4 Plattform og programvare

Følgende plattform og programvare ble benyttet på dette testoppsettet (tabell 8):

Utstyr	**Versjon**
Plattform og klient OS	Windows 7 64 bit PRO SP1
Maskinvare	Intel Core 2 CPU 2,4 GHz, 8 GB RAM
VMware	Versjon 7.0
Ubuntu VM	Ubuntu 13.0.181
vmcloud	Versjon 0.5.1
Eclipse Java EE IDE	Release 2
Wireshark	Versjon 1.6.7
vIOS	15.4

Tabell 8: Maskin og programvare versjoner i bruk på dette testoppsettet.



Figur 19: Testoppsett – plattform arkitektur.

Jeg valgte å bruke programutviklingsverktøyet Eclipse på den samme virtuelle maskinen som kontrolleren. Jeg ville elimenere eventuelle feilkilder når jeg validerte og testet egen programmeringskode. Figur 19 viser mitt testoppsett og arkitekturen på min plattform.

7.5 Implementasjon av Cisco onePK

Under følger min implementasjon av Cisco onePK i testoppsettet. Jeg beskriver først overordnet min SDN arkitektur, deretter tar jeg for meg applikasjonene og funksjonalitet som er utviklet i forhold til scenarioene til TKN beskrevet i kapittel 5.

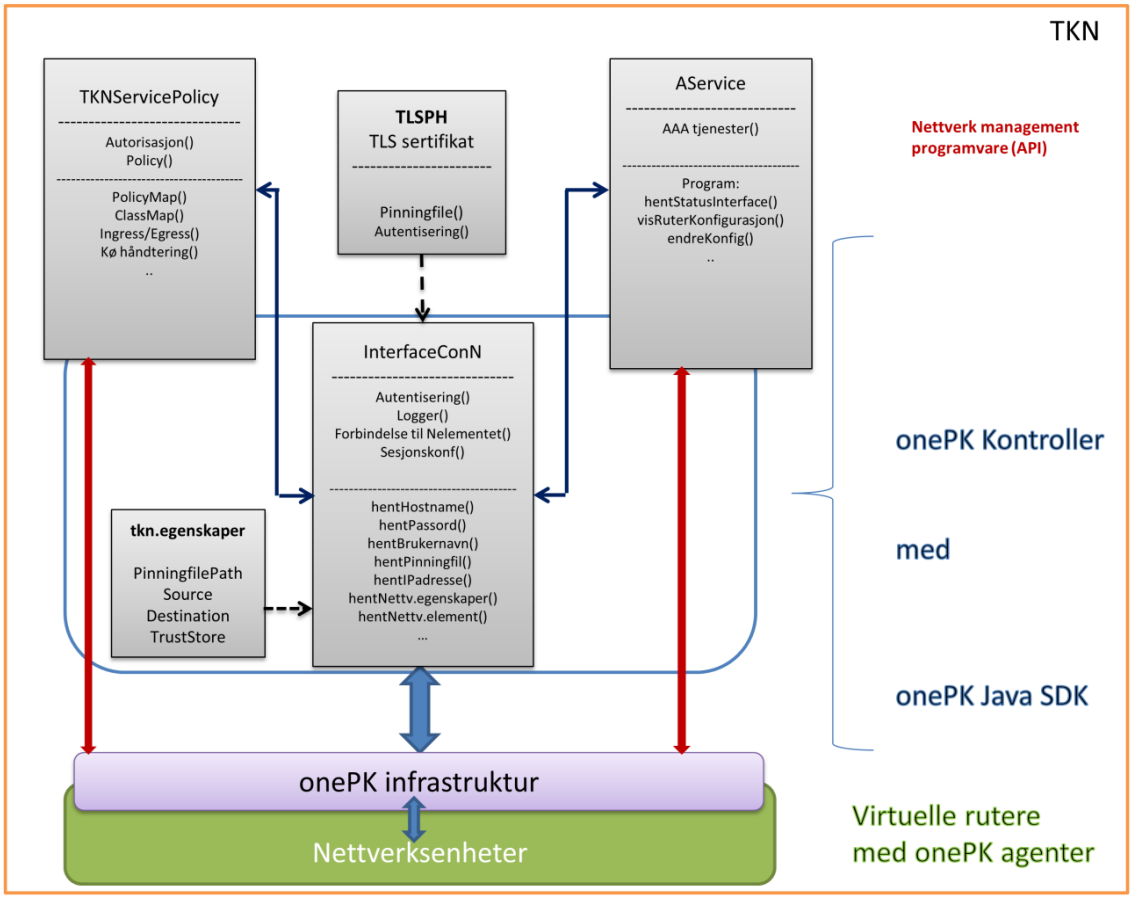
7.5.1 TKN SDN kontroller

For å kunne systematisere problemstillingene i denne oppgaven beskrev jeg to scenarioer (ref. Kapittel 5), deretter strukturerte jeg de om til spesifikk funksjonalitet og kartla hvilke tjenesteområder jeg hadde behov for å bruke.

Jeg ønsket å lage generiske metoder slik at alle applikasjonene mine kunne bruke de, i tillegg blir det enklere å implementere ny eller bedre funksjonalitet. I figur 20 har jeg laget en skisse som viser hvordan kontroller funksjonaliteten fungerer og hvordan applikasjonene kommuniserer med den. Figuren viser kun noen utvalgte metoder i klassene.
Ref. Kapittel 7.1 består onePK kontrolleren av et tjener-klient paradigme, der tjeneren befinner seg som et lag (en agent) på toppen av nettverksenheten i maskinvaren. Min onePK JAVA kode kommuniserer med tilsvarende tjenestebibliotek i maskinvaren (ref. Figur 16).

En kjernekomponent i TKN kontrolleren er klassen *InterfaceConN*. Dette er en generisk klasse som kommuniserer ned (sørgående grensesnitt) i infrastrukturen. Den inneholder mange metoder som nettverk management applikasjonene *TKNServicePolicy* og *AService* kan benytte seg av.
InterfaceConN sørger også for å etablere en sikker forbindelse mellom nettverksenheten (sørgående grensesnitt) og programmet (nordgående grensesnitt). Dette gjør den ved hjelp av *TLSPH* klassen. Rød pil i figuren, som indikerer sikker kommunikasjon[16], er først mulig etter *InterfaceCoN* har først opprettet den. Sikkerhetsfunksjonaliteten er beskrevet nærmere i kapittel 7.5.3 og 7.5.4.
TKNServicePolicy består av en applikasjon for distribusjon av en QoS policy som også er definert i klassen. QoS policy er basert på STANAG 4711. *TKNServicePolicy* er beskrevet i kapittel 7.5.5.
AService klassen består av AAA-tjeneste funksjonaliteten. I tillegg er det en liten applikasjon som tester denne tjenesten. *AService* er beskrevet i kapittel 7.5.4.

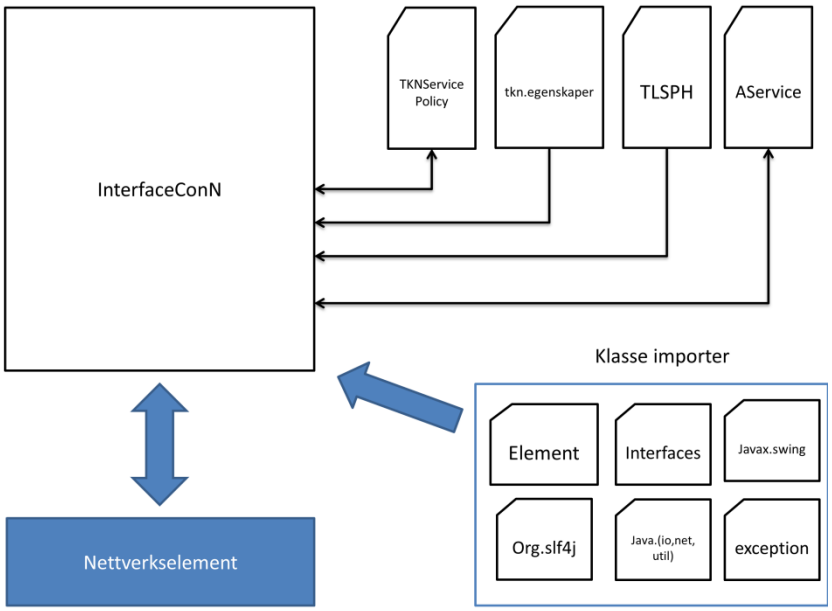
[16] Integritets- og konfidensialitetsbeskyttelse.



Figur 20: Min onePK kontroller implementasjon med noen utvalgte metoder.

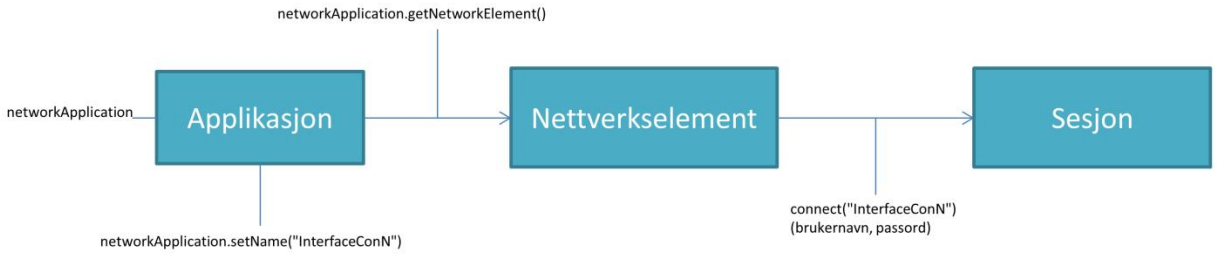
I denne oppgaven benyttet jeg meg av *End-node-hosting* distribusjon (ref. Kapittel 7.2.2) av applikasjonene. Det er en distribusjonsmetode som er best tilpasset TKN som har alt fra små til store plattform installasjoner.

7.5.2 InterfaceConN



Figur 21: InterfaceConN klassen overordnet og grunnleggende elementer.

InterfaceConN (figur 21) er en generisk klasse bestående av mye funksjonalitet. Primært har klassen funksjonalitet fra tjenesteområdet *Element Service Set* som er en kjernekomponent i onePK (ref. Tabell 7). *InterfaceConN* sitt primære ansvar er kommunikasjonen til nettverkselementet, forbindelsen (sesjonen) blir styrt av den (figur 22) slik at andre applikasjoner aksessere nettverkselementet.



Figur 22: Forbindelse til nettverkselement.

Forutsetningene for at applikasjonen skal kunne kommunisere med en nettverkskomponent må den først tilknytte seg nettverkselementet ved hjelp av nodens navn eller en IP-adresse. Dette bestemmes når man kjører applikasjonen.
I tabell 9 har jeg listet noen funksjonaliteter som ble implementert for *InterfaceConN*, resten er beskrevet i vedlegg H:

Funksjonalitet	**Beskrivelse**
Parser opsjoner.	Sjekker opsjonene til programmet: Brukernavn, passord, node navn, evt. sertifikat (TLS). Hvis bruker ikke benytter noen opsjoner benyttes en standard fil, *tkn.egenskaper* som blant annet spesifiserer et grensesnitt (for test formål).
Autentiseringsdialog.	Viser en dialog boks for autentisering av bruker. Bruker må oppgi brukernavn og passord.
«Pinning» filområdet.	Implementasjon av «TLS Certificate pinning[17]» Hvis TLS skal benyttes blir også sertifikater sjekket. Dette håndteres i klassen *TLSPH* som *InterfaceConN* kommuniserer med.
Hente status på grensesnitt.	Lister ut status fra alle grensesnitt på et nettverkselement.

Tabell 9: InterfaceConN funksjonalitet.

InterfaceConN importerer også mange andre bibliotek, deriblant:

- Org.slf4j, for [73] logging til konsoll vindu eller brukergrensesnittet ved kjøring av Java koden.
- Javax.swing, for utvikling av grafisk grensesnitt.
- onePK klassene Element og Interfaces fra onePK biblioteket.

InterfaceConN sørger for integritets- og konfidensialitetsbeskyttelse på kanalen mellom applikasjonen på endesystemet og nettverkskomponenten ved hjelp av TLS (ref. Kapittel 5).

[17] Pinning er manuell autorisasjon og lagring av et pars offentlige nøkler for fremtidig kommunikasjon.

Sikkerhetsarkitektur – TLS

For å unngå at data fra onePK applikasjoner kommuniserer ukryptert over nettverket til nettverksenhetene er det viktig å sikre kommunikasjonskanalen (ref. Kapittel 4.1.5, 5.4 og 5.5). For SDN er bruk av TLS [45] [19] foreslått som standard.

For å aktivisere onePK TLS kommunikasjon må det installeres et TLS server sertifikat som normalt må være utstedt av en sertifiseringsinstans (CA).
En forenklet konfigurasjon av TLS ble implementert på testoppsettet for å forsikre om at sertifikater og nøkler blir overført og oppbevart sikkert. To alternativer sikkerhetsarkitekturer for onePK er en implementasjon av *TLS: Certificate Pinning* eller JAVA SDK: *KeyStore/TrustStore*.

Overordnet sikkerhetsarkitektur, CA, TLS og PKI (Public Key Infrastructure) er ikke tema i denne avhandlingen, nærmere beskrivelse av disse finnes her [47] (ref. Kapittel 1.3).

Pinning
Pinning er en manuell rutine for autorisasjon og lagring av pars[18] offentlige nøkler for sikker kommunikasjon og benytter seg ikke av CA for å validere sertifikatene. I et produksjonsnettverk er ikke dette anbefalt på grunn av utfordringen med distribusjon av nøkler, men forenkler håndteringen av nøkler under utvikling av applikasjoner til bruk i større produksjons- eller operative nettverk. Pinning metoden kan benyttes uavhengig av onePK SDK programmeringsspråk. Mer om metoden her [74-76].

KeyStore/TrustStore
Er en tradisjonell mekanisme i JAVA for å støtte SSL/TLS-tilkoblinger [77]. K*eyStore* lagrer private nøkler og sertifikat som korresponderer til deres offentlige nøkler når forbindelsen krever autentisering av klient/bruker. T*rustStore* lagrer sertifikater fra en tredjepart som applikasjonen kommuniserer med eller sertifikat signert fra en CA. Mer om metoden her [78].

Aktivisering av TLS
For å sette opp TLS i min implementasjon for onePK gjennomførte jeg følgende prosedyre:

- Opprettet brukere på nettverksenhetene, passordet ble kryptert.
- Konfigurerte TLS på nettverksenhetene.
  onePK applikasjonen fungerer som en klient, nettverkselementet som server. Dobbel autentisering ble benyttet på denne implementasjonen.

[18] To noder som kommuniserer med hverandre, *per to per*.

- Aktiviserte onePK TLS kommunikasjon.
  Installerte sertifikater basert på x.509 standarden, signatur algoritme SHA-1 og 2048bit RSA kryptering [47]. Den offentlige nøkkelen har 2048bit RSA kryptering (figur 23).

```
Certificate Details:
        (..)
        Validity
            Not Before: Dec 10 00:00:00 2012 GMT
            Not After : Dec 10 00:00:00 2030 GMT
        Subject:
            unstructuredName= router1
        X509v3 extensions:
            (..)

    Data:
        (..)
Signature Algorithm: sha1WithRSAEncryption
        Issuer: CN=onePK simpleCA
        (..)
Subject: unstructuredName=router1
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                 00:ad:58:1d:2a:7e:6a:f6:4c:dc:0d:cc:27:7d:73
```

Figur 23: Et utdrag av sikkerhetsparameterne på nettverkselementene.

- Brukte TLS kommunikasjon opsjoner, eller for enkelthetens skyld *tkn.egenskaper* (figur 24) for kjøring av onePK applikasjonene. Uten opsjoner bruker kontrolleren variablene fra filen ved kjøring av metoden.

```
hostname=router1
sourceAddress=192.168.40.129
destinationAddress=10.10.10.1
appName= test_app1
domain=domain1
instance=instance1
version=0.2
trustStore=/home/tkn/truststore.jks
pinningFile=/home/tkn/onep_pinning
```

Figur 24: tkn.egenskaper

Figur 25 er et utdrag fra *InterFaceConN* Java koden og viser kallet til «paser-metoden» i *TLSPH* klassen og kallet til autentiseringsdialogen, i tillegg start av TLS pinning og metoden for å liste ut alle grensesnitt på nettverksenheten.

```
(..)
public class InterfaceConN {

    private String elementHostname;
    private String brukernavn;
    private String passord;
    private NetworkElement networkElement;
        (..)

           * Påkaller baseCon via CLI

    public static void main(String args[]) {
         InterfaceConN baseCon = new InterfaceConN();

        //Hvis ikke opsjoner er gitt vil programmet be bruker autentisere seg og benytter seg
        av et std interface definert i tkn.properties.
         baseCon.parseOptions(args);
         baseCon.showAuthenticationDialog();
         try {
             if (!baseCon.connect("InterfaceConN")) {
                 System.exit(1);
             }
                       (..)

        getLogger().info("Forbindelse med følgende nettverksenhet: - " + networkElement);
        //TLS kommunikasjon mellom nettverksenheten og kontroller.
         getLogger().info("Kommunikasjon ved hjelp av - TLS");

        //SessionConf inneholder konfigurasjonen som skal benyttes for å tilknytte seg til
        nettverksenheten.
         config = new SessionConfig(SessionTransportMode.TLS);
                //Global konstant 15002. ONEP_TLS_PORT kan endres til valgfri TLS port.
         config.setPort(OnepConstants.ONEP_TLS_PORT);
         //Enable tls pinning
         config.setTLSPinning(pinningFile, new TLSPinningHandler_new(pinningFile));

                (..)

        /**
     * Lister ut alle interfacer på en nettverksenhet.
     */
        public List<NetworkInterface> getAllInterfaces() {
         List<NetworkInterface> interfaceList = null;
         try {
             NetworkElement networkElement = getNetworkElement();
             interfaceList = networkElement.getInterfaceList(new InterfaceFilter());
         } catch (Exception e) {
             getLogger().error(e.getLocalizedMessage(), e);
         }
         return interfaceList;
    }
(…)
```

(48 av 518 kodelinjer)

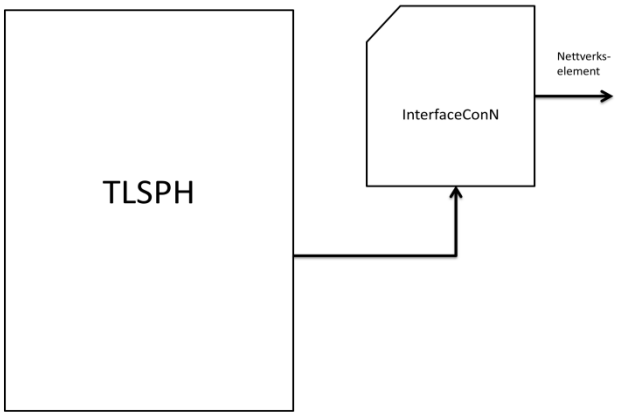
Figur 25: Utdrag av Java kode fra InterfaceConN.

Applikasjonen InterfaceConN vil også sjekke sertifikater hvis applikasjonen har aktivisert TLS. Hvis *–T* opsjonen er satt ved kjøring av applikasjonen vil kommunikasjonskanalen sikres ved hjelp av JAVA *Keystore/Truststore* metoden.
–P opsjonen vil sikre kommunikasjonen ved hjelp av *TLS Certificate Pinning* metoden.
–a definerer hvilket nettverkselement du ønsker å kjøre applikasjonen på.

Eksempel:
#java <filbane>InterfaceConN –a <IP interface/hostname> -u <brukernavn> -p <passord> **-P** <pinning fil> **-T** <TLS truststore>

For å kunne drive enklere feilsøking av applikasjonene deaktiverte jeg kryptering (TLS) i *InterfaceConN*, men aktiviserte den igjen når applikasjonene var ferdig.

7.5.3 TLSPH



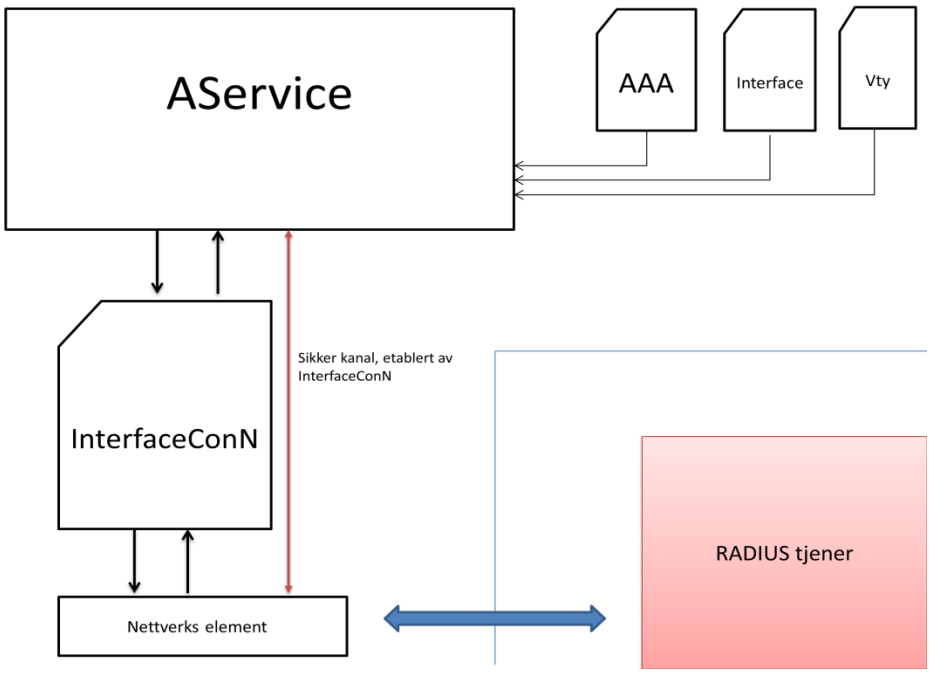
Figur 26: TLSPH klassen.

TLS-pinning handling klassen (TLSPH) (figur 26) sørger for å ivareta verifisering av TLS forbindelser som ikke er blitt verifisert fra før og beslutter hvilke sertifikater den skal akseptere eller ikke. Det er *InterFaceConN* som kaller på denne klassen.
Figur 27 viser et utdrag av Java koden og viser hvor den finner pinning filen med sertifikater og metoden for å akseptere og bruke, akseptere og bruke en gang eller avvise sertifikatet. Avvise sertifikatet terminerer også forbindelsen.

```
package com.cisco.onep.tkn;

(..)

public class TLSPH implements TLSUnverifiedElementHandler {

        String pinningFile;

        public TLSPH(String tlsPinningFile) {
                pinningFile = tlsPinningFile;

        private Decision decision = null;

        public Decision handleVerify(String host, String hashType,
                        String fingerprint, boolean changed) {
                Decision decision = showPinningDialog(host, hashType, fingerprint,
                                changed);
                return decision;
        }

}

(..)
                /**
         * Handler for TLS vertifiseringsfeil.
         *
         */
        public Decision showPinningDialog(String host, String hashType,
                                        String fingerprint, boolean changed) {
                decision = TLSUnverifiedElementHandler.Decision.REJECT;
                JPanel panel = new JPanel();
                panel.setPreferredSize(new Dimension(500, 300));
                JTextArea textArea = new JTextArea(10, 40);
                String msg = null;
                if (changed) {
             msg = " ADVARSEL: SERTIFIKATET FRA REMOTE HOST:" +host + "\n ER ANNERLEDES FRA
TIDLIGERE AKSEPTERT ";
                } else {
                        msg = "ADVARSEL: SERTIFIKATET FRA REMOTE HOST (" + host
                        + ") ER IKKE VERIFISERT.";
                }
                msg += "\n\n Hashtypen " + hashType+ " sent fra remote host (" + host+ ") er:\n"+
fingerprint;
                msg += "\n\n Du må verifiserer sertifikatet (CE) på remote host før du kan
forsette! \n";
                msg += "\n Du har følgende valg:";
                try {
                        textArea.getDocument().insertString(0, msg, null);
                } catch (BadLocationException e) {
                        e.printStackTrace();
                }
(…)

47 av 180 kodelinjer.
```

Figur 27: Utdrag av Java koden fra TLSPH.

7.5.4 AService



Figur 28: Komponentene i AService.

Ref. Kapittel 5.4.2 er det ingen SDN kontrollere som har funksjonalitet for AAA-tjenester. Men på grunn av programmerbarheten til SDN, i dette tilfelle onePK, er det mulig å lage funksjonalitet som fungerer som «*gatekeeper*» for alle brukere som ønsker å benytte seg av tjenester fra FKI, i tillegg regulere hvilke tjenester de ønsker tilgang til og hva de får lov å gjøre med tjenestene (autorisasjon).

Jeg valgte å dele sikkerhetsfunksjonaliteten i to. *InterfaceConN* klassen har fortsatt ansvaret å sikre kommunikasjonskanalen, men brukerautentisering skjer på en ekstern tjener, RADIUS (ref. Kapittel 5.4.2 og 7.5.2) [15]. Jeg oppnår da følgende:
I et sikkerhetsperspektiv så reduserer jeg sikkerhetsrisikoen ved at kontrolleren ikke har noen brukerprofiler, passord etc., i tillegg øker jeg tilgjengeligheten i nettverket ved at separate elementer i nettverket leverer ulike tjenester.
I et programmeringsperspektiv oppnår jeg et bedre objekt-orientert design som gjør det enklere å lage ny, endre eller distribuere ny funksjonalitet i nettverket.

På nettverksenhetene aktiviserte jeg RADIUS tjenesten [79], da blir nettverksenheter RADIUS klienter. Deretter utviklet jeg funksjonalitet i klassen *AService* (figur 28) for å håndtere forespørsel fra brukere og tjenestene, funksjonalitet for sikker kommunikasjon til RADIUS tjeneren, regulering av hva brukeren har tilgang til som er beskrevet i autorisasjonen sendt

fra RADIUS, og tilslutt, start og stopp av regnskapsføring som er brukerens aktiviteter på nettverket.
I scenario: *Tilgang til kjernetjenester for TKN* skal også funksjonalitet for dataflyt håndteres av kontrolleren (ref. Kapittel 5.4). I Cisco onePK kan det løses ved aktivisering av policy eller oppdatering av konfigurasjonen og eventuelle nettverksruter på nettverkselementene (ref. Kapittel 7.4.6). Tjenesteområdene *Discovery Service Set* og *Routing Service Set* har funksjonalitet for dette.

På grunn av omfanget av oppgaven og kompleksiteten utviklet jeg ikke funksjonalitet for dette. I stedet utviklet jeg for testformål et nettverks management program (API) som tester AAA-funksjonaliteten (tilgangskontroll) i kontrolleren, som også er problemstillingen definert i kapittel 1.2.
I figur 29 er et utdrag av JAVA koden for *AService* og viser metoden for autentisering av bruker.

```
public final class AService extends InterfaceConN {
       public static void main(String[] args) {
        AService legalS = new AService();

  legalS.getLogger().info("\n******* Autentiserer *******");

        //Bruker som skal skal ha tilgang til nettverkset autoriseres
        User uone = null;
        try {

            uone = new
User(legalS.getNetworkElement(),legalS.getbrukernavn(),legalS.getpassord());

        } catch (OnepException e) {
            legalS.getLogger().error(e.getLocalizedMessage(), e);
            legalS.disconnect();
            System.exit(1);
        }

        List<Attribute> Alist = null;
        try {
            Alist = uone.authenticate(null);
        } catch (OnepException e) {
            legalS.getLogger().error(e.getLocalizedMessage(), e);
            legalS.disconnect();
            System.exit(1);
        }
        legalS.getLogger().info("Autentisering OK for: " + uone.getUsername());
        legalS.getLogger().info("\n******* Henter AAA Server info *******");
        Server server = uone.getLastUsedServer();


30 av 235 kodelinjer
```

Figur 29: Utdrag av JAVA kode ASerivce. Koden viser autentisering av bruker.

FreeRadius Server

En FreeRadius tjener (figur 31) [49] ble satt opp i nettverket med det formål å drive autentisering, autorisasjon og regnskapsføring av bruker og tjenester for ulike type nettverkstilganger. Det er onePK kontrolleren (*AService*) som kommuniserer med serveren. Jeg valgte å sette opp tjeneren for å kunne teste og validere *AService* tjenesten. FreeRadius er en *deamon*[19] og kan installeres på mange ulike plattformer og kunne vært installert på Ubuntu VM plattformen som kontrolleren kjørte på.
I et distribuert operasjonelt nettverk må dette planlegges i sikkerhetsarkitekturen.

FreeRadius består flere konfigurasjonsfiler og *scripts*. På grunn av fleksibiliteten til tjeneren (kan benyttes mot mange ulike plattformer), er den tidkrevende å sette opp og tilpasse egen nettverksarkitektur. I figur 30 har jeg tatt med et utdrag av en av konfigurasjonsfilene som definerer brukerpolicyen til brukeren cops. Kontrolleren mottar denne autorisasjonsfilen og deretter regulerer tilgangen til brukeren.

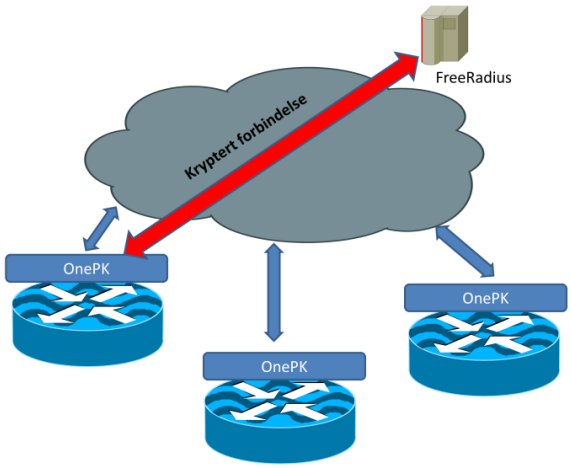
Brukeren har i dette tilfelle tilgangsnivå 15 (alle tilganger) på nettverkselementet men har kun tilgang til å kjøre applikasjonen *AService*. Det betyr brukeren har tilgang til å gjøre konfigurasjonsendringer på nettverkselementet men har ikke alle tilganger ved kjøring av tjenesten på samme nettverkselement. Med det oppnår jeg muligheten til å kunne definere teknisk operatør tilganger (driftstekniker) og andre operatør tilganger til tjenester.

Regnskap av hva brukeren foretar seg er aktivisert i hans profil. Brukeren har også begrensninger på båndbredde og databruk på to ulike grensesnitt og hvilke tiltak (forkaste datapakker) som skal gjennomføres hvis brukeren går over terskelverdiene. Funksjonalitet for å håndtere dette (de to siste linjene) er ikke implementert i *AService.*

```
cops   User-Password == "password"
       Cisco-AVPair += "priv-lvl=15",
       Cisco-AVPair += "auto-acct=enable",
       Cisco-AVPair += "allowed-app=com.cisco.onep.tkn.AService",
       Cisco-Avpair = "lcp:interface-config#1=rate-limit output 4194304 10000 10000
conform-action continue exceed-action drop",
       Cisco-Avpair += "lcp:interface-config#2=rate-limit input 3194304 10000 10000
conform-action continue exceed-action drop",

8 av 48 kodelinjer
```

Figur 30: Utdrag av en konfigurasjonsfil på FreeRadius. Filen viser brukerpolicy for brukeren cops.

[19] Deamon: Et program som kjøres som bakgrunnsprosess



Figur 31: Kommunikasjon mellom onePK og FreeRadius.

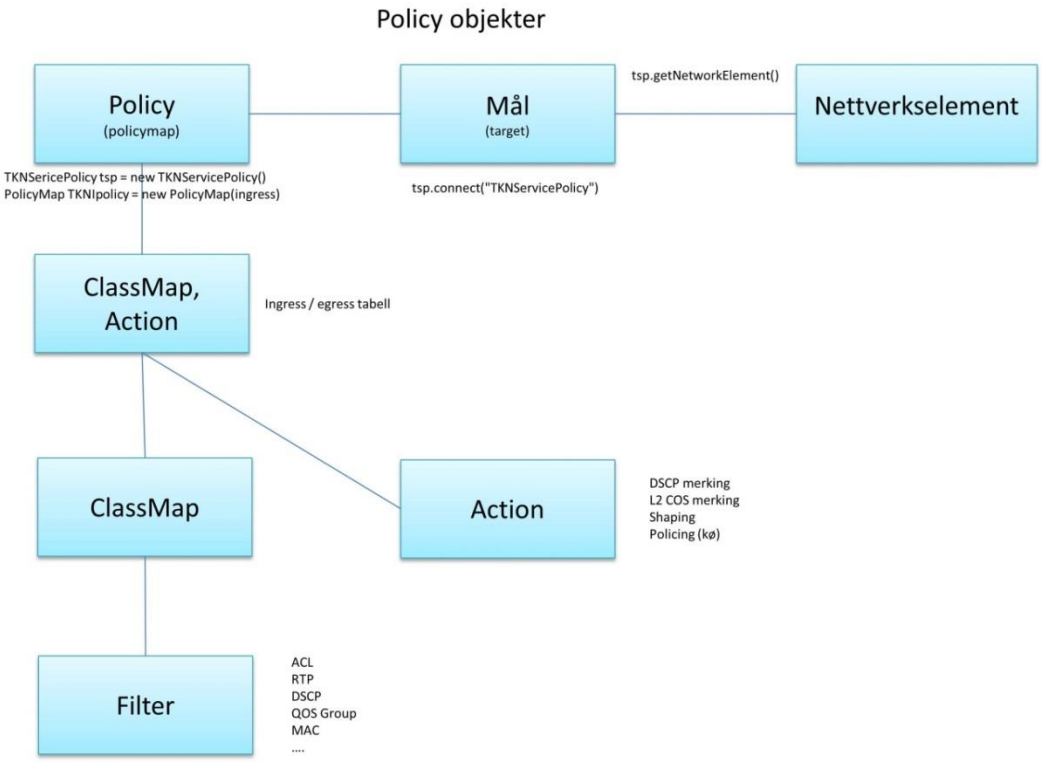
7.5.5 TKNServicePolicy

TKNServicePolicy er en nettverks management applikasjon og en generisk klasse som inneholder funksjonalitet primært fra tjenesteområdet *Policy Service Set* og Datapath Service Set. Applikasjonen kan kjøres på nettverkselementene i nettverket. Ref. Scenario: *Militær prioritet for TKN* skal kontroller kunne distribuere policy på nettverksnoder som skal fungere som IOP (ref. Kapittel 5.5). Jeg har brukt TN 1417/STANAG 4711 som grunnlag for å utvikle en policy for differensiert tjenestekvalitet for tjenester med datatrafikk som har behov for militær prioritet (ref. Kapittel 1.2).

TKNServicePolicy benytter seg av *InterfaceConN* for å etablere kommunikasjonskanalen til nettverkselementet. Autentisering kan gjennomføres med bruk av funksjonalitet fra *AService/Radius* eller av *InterfaceConN*.

Policy Service Set

Tjenesteområdet *Policy Service Set* tillater onePK applikasjoner å bruke policyer på pakker videresendt av enheten. En policy sin primærfunksjon består av å avgjøre hvilke datapakker som er relevant, deretter handle som kan bety modifisering, fjerning eller sette datapakkene i kø.



Figur 32: Policy objekter i Policy Service Set. Figuren er inspirert fra [80].

Policy Service Set har to primær Java bibliotek: *Policy* og *Policy Service*. Disse utgjør «*QoS policy*» tjenesten og datastrukturen som benyttes er: *Policy, Mål, ClassMap, Filter* og *Action* (figur 32). Jeg forklarer kort prinsippene under, utfyllende informasjon finnes i JAVA SDK som er tilgjengelig her [80].

Policy objektet:

Dette er topp nivå objektet i hierarkiet av objekter som definerer ClassMap, Filter og Action objektene slik at QoS handlinger (policy) kan defineres på nettverkselementene. Det er mulig å definere flere handlinger på et grensesnitt.

Mål objektet:

Dette er ressursen på nettverkselementet som handlingen skal defineres på. Objektet representerer et grensesnitt på et nettverkselement og hvilken retning, ingress eller egress[20].

ClassMap objektet:

Flere *ClassMap* objekter kan bli definert inni et *Policy* objekt som medfører til at en policy kan inneholde mye funksjonalitet for et nettverkselement. Videre kan hvert *ClassMap* objekt ha en eller flere *Filter* objekter som spesifiserer pakkestrømmene. Når et av filtrene i *ClassMap* objektet blir trigget vil den assosierte *Action* i *ClassMap* objektet bli utført. En viktig funksjon i *ClassMap* objektet er kø disiplinen som blir tilknyttet et grensesnitt.

Filter objektet:

Spesifiserer hendelsene på datapakkene som blir funnet like og klassifiserer dem i pakkestrømmer. Noen eksempler på filtertyper er:

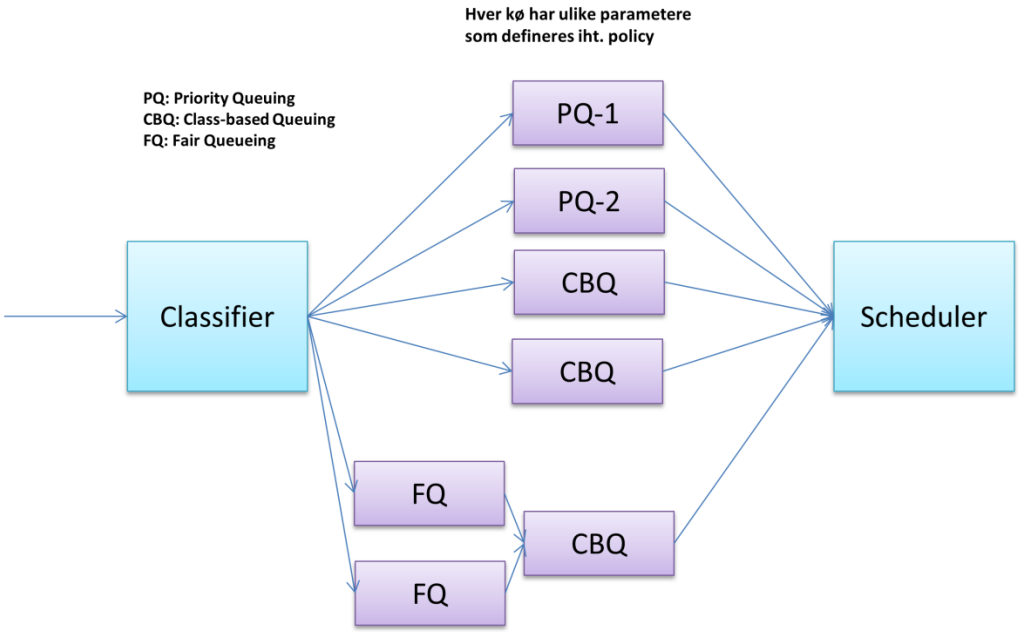
- *Access Control List* (ACL) [81].
  - Aksesskontroll lister, spesifiserer regler for porter eller IP adresser.
- *Realtime Protocol* (RTP) [82].
  - Port verdiene på RTP pakker.
- Differentiated Services Code Point (DSCP)[83].
  - Treff på DSCP feltet i IP hodet.
- L2 Class of Service (COS)[84].
  - Treff på PCP feltet i 802.1Q Ethernet rammer.

[20] Ingress er datatrafikk inn på grensesnittet. Egress er ut av grensesnittet.

Action objektet:

Spesifiserer hva som skal utføres når en viss tilstand spesifisert i *ClassMap* objektet i en spesifikk policy oppstår. Eksempler på *action* som støttes:

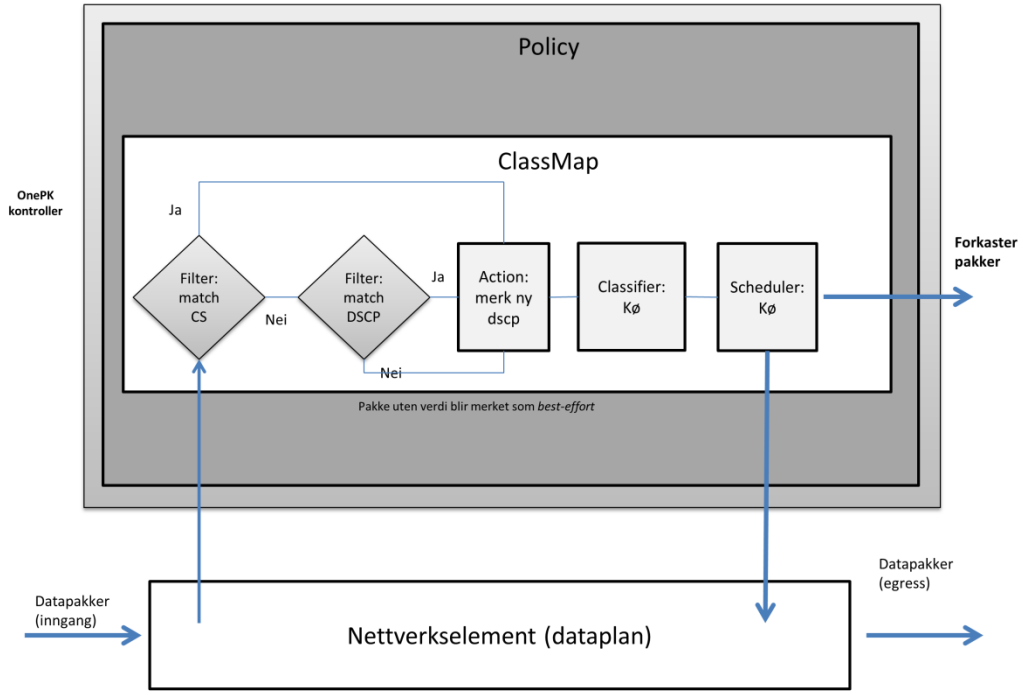
- DSCP merking
- L2 COS merking
- Kø prinsipper:
  Kø systemet består av en *Classifier*» som styrer innkommende trafikk til et sett av køer som blir håndtert videre av en *Scheduler*, figur 33 viser kø håndteringen.



PQ: Trafikk er lagt i kø basert på klassen og filtrene som er definert i policy. Planleggingen av prioritet kan være streng eller styrt. I streng prioriterings kø, er de andre køene bare planlagt når prioritetskøer er tom. I styrt prioriterings kø, er de prioriterte køer betjent før andre køene bare hvis PQ båndbredden er under en viss maksimal terskel.
CBQ: Hver kø er gitt en minimum båndbredde.
FQ: En CBQ kan ha flere køer for hver dataflyt for å gi rettferdig behandling av hver dataflyt. FQ gir ytterligere nivåer i kø hierarkiet.

Figur 33: Kø systemet. Figuren er inspirert fra [85].

Applikasjonen bruker på lik linje som *AService, InterFaceConN* til å etablere forbindelsen til nettverkselementet som deretter vil aktivisere policyen på det definerte grensesnittet. En enkel prinsippskisse for programmet vises i figur 34:



Figur 34: Prinsippskisse merking av pakker.

Datapakker som kommer inn på grensesnittet blir filtrert og sammenlignet med policyen basert på TOS verdien i IP hodet. Finner den ingen pakker fra policyen blir den behandlet som *best effort*, eventuelt finner den pakker (*match*) blir TOS verdien erstattet av ny DSCP verdi. Deretter blir den sendt til metoden *Classifier* som plasserer den i korrekt kø i henhold til prioriteten av pakken. En *Scheduler* håndterer datapakker som har høyere presedens enn annen trafikk og har ansvaret for å fjerne pakker som ikke har militær prioritet hvis kapasitet når en terskel verdi.

Datapath Service Set

Applikasjonen har behov for funksjonalitet fra tjenesteområdet *Datapath Service Set*. Tjenesteområdet gir mulighet for en applikasjon å koble seg inn i pakkestrømmen og trekke ut datapakker. Datapakkene kan enten bli kopiert til applikasjonen, fjernet eller avledet fra hvor den opprinnelig skulle. *Scheduler* metoden skal hvis kapasiteten (datastrømmer på utgående grensesnittet) når en terskel (beregnet i forhold til båndbredden), fjerne pakker som ikke har militær prioritet.

Utfordringer ved implementasjon

Under utviklingen av applikasjonen fikk jeg noen utfordringer med JAVA koden. Metoden min klarte å lese datapakkene og *«mappe»* pakker i henhold til definert policy, jeg klarte også implementerer ulike køer (ref. Kapittel 5.5 og 7.5.5). Men jeg fikk hele tiden feil i algoritmen når jeg prøvde å forkaste datapakker fra datastrømmen.
Jeg prøvde veldig mange ulike algoritmer, forskjellige type køer, jeg studerte også nærmere versjoner av onePK JAVA SDK opp mot versjoner av IOS som ikke gav noen resultater. Jeg tok tilslutt kontakt med «Cisco Dev Team» på deres utvikler forum. «Cisco Dev Team» anbefalte meg å skrive om funksjonaliteten til C språk ettersom JAVA SDK hadde foreløpig ikke støtte for funksjonalitetene i tjenesteområdet *Datapath service set*. Jeg måtte også skrive om funksjonalitet som hadde avhengigheter til klassen min.
Samtlige klasser måtte med andre ord omskrives til C språk, det arbeidet vurderte jeg som for omfattende i forhold til tidsrammen min. Min kompetanse på C var heller ikke den beste, så jeg valgte derfor å fortsette så langt jeg kunne.
I figur 35 vises et utdrag av JAVA koden for *TKNServicePolicy*. Den viser følgende:

- Opprettelsen av en policy på et nettverkselement.
- QoS tabell på ingress grensesnittet.
- Litt av *Classmap* objektet for *mapping* av DSCP verdier.
    - Match av trafikk basert på DSCP-verdi.
    - Ny «merkelapp» (DSCP-verdi) ved match.
    - *Class-maps* som blir lagt til policyen.
- Policy som blir lagt til nettverkselementet.
- Definering av et mål (grensesnitt) på nettverkselementet som policy blir aktivisert på.

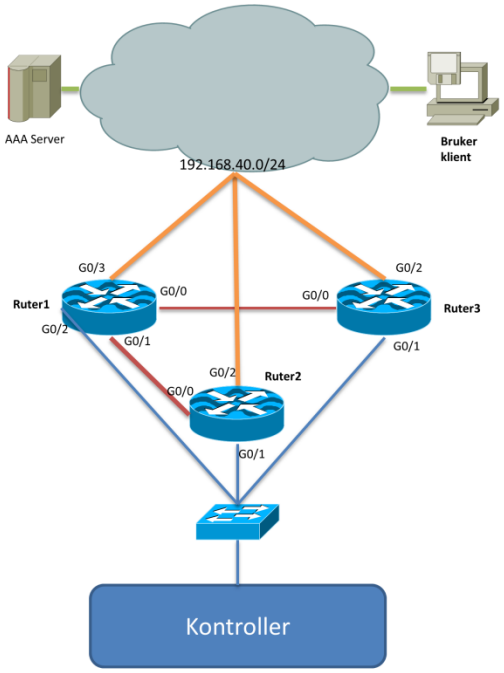
```
package com.cisco.onep.tkn;
(..)
public class TKNServicePolicy extends InterfaceConN {

    public static void main(String args[]) {
    TKNServicePolicy tsp = new TKNServicePolicy();

    //Autentisering og valg av nettverkselement.
    tsp.parseOptions(args);
    tsp.showAuthenticationDialog
(..)
        //Lager policy.
        //Bulkservice er en generisk klasse topnivå for Policy APIer.

        //Policy på nettverkselementet.
        BulkService policy = new BulkService(tsp.getNetworkElement());

        //PolicyCapabilites, mulighetene for nettverkselementet.
        PolicyCapabilities capabilities = policy.getPolicyCapabilities(true);
(..)
        //Returnerer en liste av tabeller.
        tsp.getLogger().debug("\tPolicyCapabilities - " + capabilities);
         PolicyCapabilities.TableCapabilities ingresstable
       =capabilities.getTableCapability(

       PolicyCapabilities.TableType.ONEP_POLICY_TABLE_TYPE_QOS_INGRESS)
                                .get(0);
(..)
    ClassMap TKNmap0 = new ClassMap(ingresstable);
    TKNmap0.setMatchAll(false);
    TKNmap0.addMatch(new Match.DSCP(OnepDscp.ONEP_DSCP_AF11));
(..)
    ClassMap TKNmap1 = new ClassMap(ingresstable);
    TKNmap1.setMatchAll(false);
    TKNmap1.addMatch(new Match.DSCP(OnepDscp.ONEP_DSCP_AF12));
(..)
   // Oppretter policy-map(s):
    PolicyMap TKNIpolicy = new PolicyMap(ingresstable);
(..)
    // Legger class-maps i policyen, deretter action (hva som skal gjøres).
    PolicyMap.Entry entry0 = TKNIpolicy.createEntry(TKNmap0);
    entry0.addAction(new Action.Mark(MarkType.MARK_IP_DSCP, 10));
    PolicyMap.Entry entry1 = TKNIpolicy.createEntry(TKNmap1);
    entry1.addAction(new Action.Mark(MarkType.MARK_IP_DSCP, 12));
(..)
//Legger til policy-map(s) til nettverkselementet. (ingress)
    tsp.getLogger().info("Legger til PolicyMap...");
    List<PolicyMapResult> poresult = policy.submitPolicyMap(TKNIpolicy);
    for (PolicyMapResult pmr : poresult) {
             tsp.getLogger().info("PolicyMapResult code (0 hvis alt ok) - " +
pmr.getResultCode());
              tsp.getLogger().info("PolicyMapResult message (hvis noe) - " +
pmr.getResultMessage());
              tsp.getLogger().info("PolicyMapResult status (top level status) - " +
pmr.getStatus());
(..)
              for (EntryResult entryRes : pmr.getEntryResultList()) {
                    tsp.getLogger().info(entryRes.toString());
  (..)
  // Lager mål
      Target target = new Target(nwInterface, TargetLocation.HARDWARE_DEFINED_INPUT);
(…)
59 av 268 kodelinjer
```

Figur 35: Utdrag av JAVA koden fra TKNServicePolicy.

7.5.6 Nettverkstopologi

Jeg ønsket å lage en nettverkstoplogi som passet scenarioene (ref. Kapittel 5), i tillegg må Cisco onePK kontrolleren kunne kommunisere med alle nettverkselementene i topologien. Testoppsettet ble derfor satt opp med tre rutere, en sentral kontroller og 6 ulike nettverk. Dette gir mulighet for å kjøre applikasjonene og implementere funksjonalitet på alle ruterne på alle grensesnitt. Alle linkene ble som standard satt til 100 Mb/s.
Alle ruterne hadde en standard konfigurasjon ved oppstart av nettverkstopologien. Dette gjorde det mulig og «nullstille» nettverkstopologien hvis testene eller programmene skulle feile og feilsøking skulle bli vanskelig. Vedlegg C viser et utdrag av nettverkstopologi definisjonsfilen for testoppsettet.

Nettverket i dette oppsettet (figur 36) så slik ut:



Ruter	Interface	IP addresse
Ruter 1	G0/0	10.10.20.110/24
	G0/1	10.10.30.110/24
	G0/2	10.10.10.110/24
	G0/3	192.168.40.129/23
Ruter 2	G0/0	10.10.20.120/24
	G0/1	10.10.10.120/24
	G0/2	192.168.40.130/24
	G0/3	Reservert
Ruter 3	G0/0	10.10.30.130/24
	G0/1	10.10.10.130/24
	G0/2	192.168.40.131/24
	G0/3	Reservert
Kontroller	Eth0	10.10.10.1
Bruker klient	Eth0	192.168.1.160/24
AAA Server	Eth0	192.168.0.106/24

Figur 36: Nettverkstopologi.

Fra terminal vindu i Ubuntu kan nettverket etableres ved hjelp av kommandoen vmcloud.
#vmcloud [delkommando [<delkommando alternativer>]]

Mest brukte delkommandoer:

netcreate	Lager nettverk fra topologi definisjonsfilen
netlist	Lister nettverkstopologien og status
netdelete	Sletter eksisterende nettverk
netstart	Starter nettverket fra "shutdown"
-v FILE	Spesifiserer hvilken xml nettverkstoplogi fil.

Konfigurasjonen ble skrevet i følgende filer:

router1.con
router2.con
router3.con

Et utdrag fra koden vises i figuren 37, koden er standard Cisco konfigurasjonskode.

```
version 15.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Router1
(..)
interface GigabitEthernet0/0
 ip address 10.10.20.110 255.255.255.0
 duplex auto
 speed auto
 no shutdown

11 av 114 kodelinjer (..)
```

Figur 37: Utdrag fra konfigurasjonskode på nettverkselementet.

7.6 Oppsummering implementasjon

Dette kapittelet har vært en kort gjennomgang av noen SDN kontrollere og en mer utfyllende beskrivelse av kontrolleren Cisco onePK som ble valgt for denne oppgaven. Kapittelet har også presentert et testoppsett bestående av nettverksemulering, virtuelle maskiner (VM) og en nettverkstopologi som testene kan gjennomføres på.
Videre har kapittelet beskrevet Cisco onePK kontroller funksjonaliteten og applikasjonene som ble utviklet i programmeringsspråket JAVA. Utfordringer og kompleksiteten av implementasjonen har også blitt synliggjort.

Neste kapittel i denne oppgaven tar for seg testene som ble gjort og vurderingene av dem.

8. Test, validering og resultater

I dette kapitelet beskrives testene som ble gjort i oppsettet for å validere nettverket. Deretter blir testene, resultatene og konklusjonen for hvert scenario beskrevet. Tilslutt kommer mine anbefalinger for videre arbeid.

Det er vanskelig å visualisere programmeringskode for nettverksfunksjonalitet, ruterne viser heller ikke hva som skjer i sine brukergrensesnitt. Jeg har derfor valgt å vedlegge en del utdrag fra logger og utskrift fra konsollvinduet ved kjøring av applikasjonene.

8.1 Introduksjon

Ref. kapittel 7.5.6 ble nettverket ble emulert ved hjelp av `vmcloud` og ble brukt for alle testene i dette testoppsettet. JAVA koden ble kompilert og kjørt i Eclipse. For å kunne se sammenheng mellom applikasjonene og det som skjer i nettverket ble konsoll vinduet i Eclipse benyttet flittig. Nettverksprotokoll analyse verktøyet Wireshark ble benyttet for å analysere trafikkstrømmene, spesielt interessant var strømmen på nordgående grensesnitt mellom applikasjonene og kontroller.
Feilsøkingsverktøyet i onePK ble også benyttet for å analysere strømmen og for feilsøking (Figur 38).

```
Router1# debug onep server interface <-- enable onep server interface debugging
*Mar  9 13:35:52.037: [debug][tid: 47] [session id: 7061] [ONEP][Policy]: [FLOW] Session exit 7061
[onep_policy_session_exit:792]

*Mar  9 13:35:52.037: [debug][tid: 47] [session id: 7061] [ONEP][Policy]: [FLOW] Deleting Classes
[onep_al_policy_delete_all_classes:617]

*Mar  9 13:35:52.038: [debug][tid: 47] [session id: 7061] [ONEP][Policy]: [FLOW] Deleting Class=1
[onep_al_policy_delete_all_classes:622]

*Mar  9 13:35:52.038: [debug][tid: 47] [session id: 7061] [ONEP][Policy]: [FLOW] Deleting Class=2
[onep_al_policy_delete_all_classes:622]

*Mar  9 13:35:52.038: [debug][tid: 47] [session id: 7061] [ONEP][Policy]: [FLOW] onep_class->num_filters
0 [onep_al_policy_delete_all_classes:634]
```

Figur 38: Feilsøking i onePK.

8.2 Nettverkstopologi og RADIUS

8.2.1 Beskrivelse

Vmcloud leser flere XML-scripts deretter lastes konfigurasjonsfilene på ruterne (ref. Kapittel 7.5.6). Nettverket testes ved hjelp av to diagnostikk verktøy i vmcloud: netlist og netdiag. Ping kommandoer mot grensesnittene ble deretter benyttet for å teste linkene og rutingen i nettverket.

8.2.2 RADIUS tjener

RADIUS tjeneren har stor kompleksitet på grunn av tilpasningen som kan gjøres for at den skal kunne fungere sammen med stort antall systemer og ulike nettverkstopologier. Tjeneren anbefales å kjøres i *debug mode*. Gjøres det en endring i en av konfigurasjonsfilene på tjeneren er det ikke mulig å se om den fungerer som tiltenkt hvis den ikke først testes i *debug mode*.

For å starte den i debug mode:
`#radiusd –X`

8.2.3 Resultat

Utdrag fra `netdiag` finnes i vedlegg E.
Under vises deler fra `netlist`**:**

```
cisco@onepk:~$ vmcloud netlist
List network topologies
Preconfigured Sample Networks:

3node

/usr/3node/3node.virl

Networks Running On The System:

Name                 State                                Routable LAN
3node                Active                               10.10.10.1/24
```

Utskrift fra `debug mode` fra RADIUS kan sees i vedlegg F: RADIUS (side 154).

8.2.4 Konklusjon

Nettverket fungerte som forventet, men emulering av nettverkstopologien på en virtuell maskin er ressursintensivt. Selv med 8 GB intern minne måtte jeg først kjøre topologien før jeg kunne starte Eclipse. Topologien feilet ved lasting av vIOS hvis Eclipse allerede var startet på grunn av mangel på tilgjengelig minne.
Videre er det en nødvendighet å skissere nettverket og spesifiserer detaljene i topologien før man begynner å skrive scriptene. Endringer i ettertid tar tid på grunn av endringene som må gjøres i mange filer.

8.3 Tester for scenario: Tilgang til kjernetjenester for TKN

8.3.1 Beskrivelse

For testene opprettet jeg tre brukere, *cops* med begrensede rettigheter i applikasjonen AServirce, *uone* med alle rettigheter og *testuser* som ikke har autorisasjon til å kjøre AService. Forsøkene skulle:

- Teste kommunikasjonskanalen mellom applikasjon (bruker grensesnitt), kontroller og nettverkselementet.
- Teste hvordan onePK håndterer funksjonalitet for AAA-tjenester i et nettverk.
- Verifisere at forbindelsen blir terminert når bruker er ferdig å bruke sine tjenester og avslutter.
  Regnskapet skal kunne hentes fra RADIUS.
- Teste hva som skjer når det oppstår brudd i kommunikasjonskanalen mellom applikasjon og nettverkselementet.

8.3.2 Resultater

Noen resultater fra kjøring av *AService* er vist i vedlegg F. Under viser konsoll vinduet at autentiseringssekvensen har vært vellykket og sikker kommunikasjonskanal er etablert:

```
187  [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Kommunikasjon ved
hjelp av - TLS
177132 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Forbindelse
etablert –
```

Dialogboksen brukeren får i sitt display er vedlagt i vedlegg F.

onePK kommuniserer med RADIUS for autentisering av bruker, deretter applikasjonen, tilslutt mottas autorisasjonen til brukeren (vedlegg F: Test 1). Under ser vi forespørselen fra brukeren loggført på FreeRadius tjeneren:

```
Packet-Type = Access-Request
Mon Mar  2 12:02:51 2015

  User-Name = «uone»
  User-Password = «h3mmelig»
  NAS-IP-Address = 192.168.40.144
```

```
Client-IP-Address = 192.168.40.144
```

Testene viser videre at:

- Brukeren *uone* får tilgang og mulighet til å kjøre alle valgene i programmet mot nettverkselementet (vedlegg F: Test 2).
- Brukeren *cops* fikk tilgang med begrensninger i henhold til spesifikasjonen i policyen (vedlegg F: Test 3).
- Brukeren *testuser* som har tilgangsrettigheter til nettverkselementet men ikke nok til å kjøre applikasjonen *AService*, får ikke tilgang til nettverkselementet (vedlegg F: Test 4).

Som siste test blir kommunikasjonskanalen brutt og hendelser observert. Vedlegg G: Test 5, viser utskrift fra kontroller i tillegg ble kanalen observert fra Wireshark.

Vedlegg G viser også hva RADIUS returnerer under *debug mode* og utskrift av logger når *uone* kjørte *AService*.

8.3.3 Observasjoner

Under Test 5 ved brudd i kommunikasjonskanalen ble det gjort noen interessante observasjoner. Ved mangel av SYNC-ACK[21][85] mellom nettverkselementet og kontroller ble prosessen på nettverkselementet avsluttet etter 60 sekunder og det oppstod brudd i forbindelsen. *Keepalive* parameteren var som standard satt til 60 sekunder (vedlegg F: Test 5). Jeg undersøkte nærmere for å se om det gikk an å slå av *keepalive* eller programmere inn andre parametere. Parameterne som kunne endres er listet i vedlegg F: Parametere.

Ved kjøring av prosessene ble også observasjoner av prosessor (CPU) og minneforbruk gjort. Vedlegg F: Test 6 viser utskrift fra ruteren.

8.3.4 Konklusjoner

AAA funksjonalitet

Testene i testoppsettet viser at onePK kan benyttes til å håndtere funksjonalitet for AAA tjenester i et nettverk: 1) registrere og autentisere nettverkselement; 2) autentisere brukere;

[21] SYNC/ACK (RFC 5246) blir benyttet ved etablering av SSL/TLS forbindelser og er en del av «handshake» prosedyren mellom tjener og klient.

3) håndtere autorisasjon. Funksjonaliteten kan programmeres etter behov og onePK kan lese hele TCP/IP modellen, funksjonaliteten kan derfor utvides til: 4) autentisering av noder og binde de til porter; 5) håndtering av dataflyt og bruker/node mobilitet.

Vanlige aksess kontroll metoder i tradisjonelle nettverk har begrensninger når brukeren er mobil. Hvis funksjonaliteten håndteres av kontrolleren reduseres kompleksiteten i nettverket (eksempel: MAC aksess lister på svitsjer), det blir enklere å implementere eller endre funksjonaliteten og gjennomføre overvåking over nodenes bruk av kjernetjenester.

Kommunikasjonskanalen

Testene viser at brudd i kommunikasjonskanalen kan ha konsekvens for TKN med store forsinkelser eller som opplever ofte brudd i kommunikasjonsforbindelsen, forutsatt at kontrolleren befinner seg i kjernenettverket i FKI, SDN kontroller sentralisert i topologien. Håndtering av forsinkelser ved bruk av en annen SDN kontroller enn onePK og med bruk av OpenFlow er foreløpig ikke dokumentert i noen publikasjoner fra forskning. Dette kan ha sammenheng med at SDNi fortsatt eksisterer som et utkast til en standard og er ikke ferdigstilt. Dette er helt klart et viktig tema som må forskes mer på og drøftes når SDN skal vurderes for FKI og Taktiske kommunikasjonsnoder.

Med en distribuert eller hieratisk oppbygget topologi vil prosessen ha kommunikasjon til lokal onePK kontroller og vil ikke ha behov for kommunikasjon til sentral kontroller.

Applikasjonsprosessen som ble kjørt i denne testen var ikke ressursintensiv på ruteren. Utskrift (vedlegg F) fra ruteren viser lite forbruk i dette forsøket.

8.4 Tester av scenario: Differensiert tjenestekvalitet for TKN.

8.4.1 Beskrivelse

For testene ble ping kommandoer i Ubuntu med mulighet til å manipulere TOS feltet i IP-hodet benyttet for å generere trafikk. Ved å benytte TOS desimal eller hex format kunne jeg sende trafikk inn i nettverket spesifisert fra RFC 2474 (tabell 10), og ved hjelp av logger og kommandoer på ruteren kunne jeg deretter se om *mappingen* fra RFC 2474 til STANAG 4711 fungerte og ble håndtert korrekt.
Under er ett eksempel i bruk der trafikk med DSCP verdi xxx-010 skal bli behandlet som forbindelses løs FLASH[22] trafikk. Den nye DSCP/PHP klassen er AF23 (tabell 11). TOS verdien i IP-hodet endres til dette og skal ha den samme behandlingen gjennom hele nettverket.

[22] FLASH melding: Skal behandles så raskt som mulig og har høy presedens.

RFC 2474:

TOS (des)	TOS (hex)	TOS Presedens navn	TOS Presedens (des)	DSCP (bin)	DSCP/PHB Klasse
72	0x48	Immediate	2	**010010**	**Af21**

Tabell 10: RFC 2474 prioritet Immediate

ST4711:

Presedens nivå	IPMPL-DSCP (mil. prioritet)	AF verdi / DSCP/PHP	Desimal verdi
CL(forbindelsesløs)-FLASH	**xxx 010**	**AF23**	22

Tabell 11: ST4711 CL FLASH

Det ble gjennomført tester ved bruk av *Class Selector* verdier (CS) [86] og *Assured Forwarding* (AF) [86] i TOS for å se om *mapping* funksjonaliteten i kontrolleren fungerte. Kommandoen `# ping – Q <verdi> <IP-adresse>` vil generere trafikk, `-Q` opsjonen spesifiserer verdien. Ved hjelp av Wireshark, logger og kommandoer på ruteren kunne jeg gjøre analyser av testene.

8.4.2 Resultater

TKNServicePolicy programmet ble startet og kjørt mot grensesnittet G 0/3. *InterfaceConN* sørget for at forbindelsen mellom kontroller og grensesnittet ble etablert og sikker kanal opprettet. Deretter ble policy implementert av kontroller på nettverkselementets ingress grensesnitt, et grensnitt sensorene i det taktiske nettverket har som sin gateway[23] adresse:

```
Lager målet (target) på interfacet GigabitEthernet0/3
10169 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Activate Policy to
its targets...

(…)
```

[23] Gateway er nettverksadressen til grensesnittet ut av eget nettverk.

Av testresultatene kunne jeg se at TOS feltet i IP-hode ble lest og sammenlignet med policyen som ble implementert. Pakkene får deretter en ny merkelapp (TOS verdi).
Det ble også implementert en policy på nettverkselementets egress grensesnitt, trafikk som skal til FKI fra IOP. TOS feltet i IP-hode ble lest og sammenlignet med policyen som er implementert, klassifisert og lagt i kø. Under viser prioritet kø for DSCP verdi AF23:

```
//Oppretter prioritet kø for DSCP verdi AF23
entry8.addAction(new Action.PriorityQueue(500,BandwidthUnits.BW_UNITS_KBPS,
1, 10000, QueueSizeUnits.QUEUE_UNITS_BYTES));
```

Kun deler av algoritmen fungerte, algoritmens *Scheduler* hadde ingen funksjon. Alle pakkene fra ingress ble sendt ut på egress grensesnittet og motsatt, ingen pakker ble forkastet.
Prosessen kan avsluttes lokalt på ruteren slik:

```
# onep stop session <sesjons iD>
```

Når applikasjonsprosessen avsluttes blir policy fjernet fra IOP noden deretter blir forbindelsen brutt. Minne og CPU frigjøres fra nettverkselementet.
Resultater fra testene finnes i vedlegg G.

8.4.3 Konklusjoner

Policy

Testene viser at policy for differensiert tjenestekvalitet for trafikk med militær prioritet kan implementeres på en node *ad-hoc* og kan kjøres så lenge operasjonen pågår. Prosessen kan avsluttes av kontrolleren eller av en lokal operatør ved noden. Algoritmen på denne testen var ikke komplett, *Scheduler* metoden ble derfor ikke testet.
Med en komplett metode bør det genereres så mye trafikk på nettverkselementet at den oppnår terskelen for metning slik at den begynner å forkaste pakker som ikke har militær prioritet. Dette kan måles på utgående grensesnitt.

Ytelse og bruk av ressurser

Av statistikken fra nettverkselementet viste policy-prosessen seg ikke å være ressursintensiv, CPU bruken var lav og minneforbruket var kun 6000 bytes. Dette ble igjen frigjort når prosessen ble avsluttet. For Taktiske kommunikasjonsnoder er kravene til vekt, mobilitet, robusthet og stabilitet høye. Samtidig skal de kunne håndtere mye trafikk og stadig endringer i topologien. Nettverksutstyr som benyttes i strategiske og stasjonære nettverk har gode egenskaper til å kunne håndtere mye datatrafikk og ofte endringer. Komponenter

med egenskaper som du finner i stasjonære nettverk og samtidig tilfredsstiller kravene for Taktiske kommunikasjonsnoder er sjelden «hyllevare».

Ved hjelp av SDN Cisco onePK kan policyer eller annen management i nettverket bli håndtert utenfor dataplan segmentet og kan kjøres på nettverkselementet ved behov. Elementet vil ha en enklere standard konfigurasjon, nye policyer kan implementeres *ad-hoc* for hver operasjon eller ved behov. Når operasjonen eller prosessen er ferdig vil den frigjøre ressurser (CPU, minne).

8.5 Oppsummering av test, validering og resultater

Jeg gjennomførte tester basert på to scenarioer for Taktiske kommunikasjonsnoder. Målet for testene var å se om Cisco sitt SDN konsept onePK kan benyttes for å understøtte FKIs behov for autentisering av brukere og tjenester på Taktiske kommunikasjonsnoder. I tillegg, om onePK kunne iverksette forhåndsdefinerte policyer for differensiert tjenestekvalitet.

Applikasjonene og funksjonaliteten som fungerte på plattformen i testoppsettet vil kunne fungere på de andre plattformene til Cisco uten å måtte modifisere eller tilpasse koden. Dette er en stor fordel og gir derfor utviklere mulighet til å lage emulerte nettverk og utvikle funksjonalitet før de implementeres på operative nettverk.

onePK SDK har også sine begrensninger. Funksjonalitet som ikke støttes i tjenesteområdene er heller ikke mulig å utvikle på egenhånd. Cisco jobber stadig med utvidelser av funksjonaliteten i onePK. Et eksempel på dette er Cisco Development Forum der utviklere fra brukermiljøet i industrien henvender seg direkte til utviklingsteamet til Cisco. En tett integrasjon mellom brukermiljøet og utviklermiljøet kan etter min vurdering være en god forretningspolicy og gjøre Cisco til en sterk konkurrent i markedet når industri skal velge SDN teknologi for sine nettverk.

9. Konklusjon

I denne oppgaven har jeg vist Software-Defined Networking basert autentisering og aksess kontroll for Forsvarets taktiske kommunikasjonsnoder. I tillegg har jeg vist hvordan policy for differensiert tjenestekvalitet kan bli implementert i nettverket for tjenester som har behov for militær prioritet i Forsvarets kommunikasjonsinfrastruktur. Dette er blitt demonstrert i et testoppsett ved hjelp av Cisco sin SDN kontroller onePK, hvor programmer for AAA-funksjonalitet og policy for merking av trafikk ble utviklet.

Oppgaven har også gitt en beskrivelse av noen utfordringer ved bruk av SDN teknologi i FKI og mulige løsninger med bruk av SDN kontroller. Videre, at behovet for mer forskning og utvikling er nødvendig for nettverk som er dynamiske, mobile og som har utfordringer med forsinkelser og eventuelle brudd på kommunikasjonsforbindelsene.

SDN gir muligheter for å lage nye løsninger og funksjonalitet gjennom utvikling av programvare, samtidig gir den økt kompleksitet og stiller høyere krav til nettverkskompetanse for programutviklere.
Min anbefaling ved eventuell implementasjon av SDN teknologi i Forsvaret er at Forsvaret må vurdere kompetansebehovet innenfor *development and operations* (DevOps). En DevOps operatør har sitt kompetansefelt innenfor programutvikling, IT operasjoner og kvalitetssikring. Programvarebasert teknologi kommer til å øke i omfang, SDN konseptet er et eksempel på dette innenfor nettverksteknologier.

SDN er sagt å være paradigme skifte for nettverksteknologier på grunn av sin fleksibilitet ved hjelp av programutvikling. Det eksisterer fortsatt veldig få kommersielle aktører som kan levere en komplett løsning for brukerne. Det gjenstår også mye forskning og standardisering innenfor SDN.

Hybride løsninger, SDN implementasjoner i eksisterende nettverksinfrastruktur, kan etter min vurdering gi Forsvaret økt operativ effekt for Taktiske kommunikasjonsnoder på grunn av mulighetene for bedre kontroll og styring.

9.1 Videre arbeid

Gjennom arbeidet med denne oppgaven så jeg flere muligheter for forbedringer av programmene og nye interessante tester. Under har jeg listet noen av de viktigste:

- Cisco Java SDK har i denne versjonen begrenset funksjonalitet, spesielt for pakkestrømmer på nettverkselementene. Ny versjon skal være på banen, men mitt inntrykk er at funksjonaliteten utvikles først i C deretter i JAVA. Jeg anbefaler derfor å bruke Cisco C SDK.

- AAA funksjonaliteten kan utvides til å registrere og autentisere svitsjer, i tillegg binde noder til svitsjer og porter.

- *Keepalive* signalet mellom applikasjonen, kontroller og nettverkselementet var ikke mulig å slå av. Det er mulig å endre en del av parameterne slik at ikke nettverksforbindelsen blir brutt og prosessen avsluttet etter kun 60 sekunder. Jeg ville prøvd å utvikle en metode for å komme rundt problematikken.

- Det ble ikke gjennomført tester på operative kommunikasjonsnoder. Resultatene fra emuleringen var uansett relevante og interessante og danner er et godt grunnlag for eventuelle operative tester.

- Testoppsettet i denne størrelsesorden kunne kjøres på en stasjonær maskin med minimum 8 GB internminne. Jeg anbefaler å doble internminne og benytte quad-core[24] prosessor arkitektur. Emulering av nettverkselementer og bruk av vIOS er en ressursintensiv prosess.

- Med ny versjon av JAVA SDK bør policyprogrammet og algoritmen for merking og kø håndtering ferdigstilles. Den bør deretter testes ved å sende nok data inn i nettverket slik at nettverkselementet må iverksette *minimum mode* og kun forkaste datapakker som ikke har militær prioritet (ref. Kapittel. 5.5).

- Jeg anbefaler Forsvaret å se nærmere på STANAG 4711 og bruken av den på noder som fungerer som IOP i FoN og på TKN i FKI. Min vurdering er at standarden kan benyttes på nasjonale IOP noder og på noder som fungerer som IOP i FoN. Forutsetningen er at Forsvaret spesifiserer hvilke tjenester som har behov for militær prioritet og lager et konsept for merking av tjenestene i FKI. Dette bør synliggjøres i en CONOPS eller i en arkitekturbeskrivelse for FKI.

[24] Prosessor arkitektur som er designet for å håndtere store kalkulasjoner og virtualisering.

Liste med akronymer

3G	Tredje generasjons
AAA	Authentication, authorization, accounting
ACL	Access Control List
API	Application Programming Interface
ASN	Autonomous system number
ATM	Asynchcronous Transfer Mode
BGP	Border Gateway Protocol
CA	Certifate
CBQ	Class-based Queuing
CCNA	Cisco Certified Network Associate
CLI	Command Line Interface
CONOPS	Concept of operations
COS	Class of Service
CYFOR	Cyberforsvaret
DevNet	Cisco Development Network
DevOPS	Development and operations
DiffServ	Differentiated Services
DoS	Denial of Service
DSCP	Differentiated Services Code Point
EGP	Exterior Gateway Protocol
FD	Forsvarsdepartementet
FFI	Forsvarets forskningsinstitutt

FKI	Forsvarets kommunikasjonsinfrastruktur
FOH	Forsvarets hovedkvarter
FoN	Federation of Networks
FQ	Fair Queueing
IGP	Interior Gateway Protocol
IKT	Informasjons og kommunikasjonsteknologi
IntServ	Integrated Services
IOP	Interoperability Point
IP	Internet Protocol
K2IS	Kommando, kontroll og informasjonssystem
MAC	Media Access Control
MANET	Mobile Ad-hoc Network
MPLS	Multiprotocol Label Switching
MTKN	Middels stor taktisk kommunikasjonsnode
NATO	North Atlantic Treaty Organization
NSA	NATO Standardization Agency
OF	OpenFlow
ONE	Open Network Environment
OS	Operativ system
PK	Platform Kit
PQ	Prority Queue
QoS	Quality of Serivce
RADIUS	Remote Authentication Dial-In User Service

RIB	Routing informationBase
RSVP	The Resource Reservation Protocol
RTP	Realtime protocol
SATCOM	Satellite ommunications
SDK	Software Devolopment Kit
SDN	Software Defined Networking
SLA	Service Level Agreements
SSL	Secure Sockets Layer
STANAG	Standardization Agreement
TACOMS	Tactical Interoperable Communications Standards
TCP	Transmission Control Protocol
TKN	Taktisk kommunikasjonsnode
TLS	Transport Layer Security
TN	Technical note
TOS	Taktisk område samband
TOS	Type Of Service
UDP	User Datagram Protocol
UHF	Ultra High Frequency
VLAN	Virtual Local Network
VLAN PCP	Virtual Local Area Network Priority Code Point
VLL	Virtual Leased Lines

VM	Virtuell maskin
Vmcloud	Cisco Network Device Emulator
VPLS	Virtual Private Lan Services
VPN	Virtual Private Networks
WRED	Weighted Random Early Discard

Referanser

[1] Telenor, “Refarming and other challenges with mobile communications,” Report19 april 2012 2012.

[2] Cisco, “Cisco Visual Networking Index: Global Mobile Data Traffix Forecast Update, 2013-2018,” Cisco2014.

[3] D. Belson, “The State of the Internet,” Akamai - Faster Forward, Cambridge, Massachusetts2013.

[4] B. Salisbury. (2013, Januar). *Inside Google’s Software-Defined Network*. Available: http://www.networkcomputing.com/networking/inside-googles-software-defined-network/a/d-id/1234201?

[5] Samferdselsdepartementet, “Bredbånd til alle - rammevilkår for auksjon av frekvensene i 800 Mhz-båndet,” ed. Regjeringen.no: Regjeringen Stoltenberg II, 2013, p. 1.

[6] O. R. Valmot, “Snart skjer det noe drastisk med mobilsamtalene dine,” ed: Teknisk Ukeblad, 2015, p. 1.

[7] J. E. o. O. I. B. Voldhaug, “FFI rapport 2013/01410 - Kapasitetsbehov i Forsvarets kommunikasjonsinfrastruktur (BEGRENSET),” Forsvarets forskningsinstitutt, Kjeller2013.

[8] Forsvaret, “Vedlegg A - Kravdokument Taktiske Kommunikasjonsløsninger (BEGRENSET),” Forsvaret09/2013.

[9] O. I. Bentstuen, “Definisjoner innen Forsvarets kommunikasjonsinfrastruktur – innspill til diskusjon,” Forsvarets forskningsinstitutt, Kjeller2009.

[10] I. M. Jeroen Hoebeke, Bart Dhoedt and Piet Demeester, “An overview of Mobile Ad Hoc Networks:Applications and challenges,” INTEC Ghent University, Ghent2012.

[11] NATO, “STANAG 4711 - Interoperability point Quality of Service (IOP QoS) (NATO Unclassified),” NATO, Haag01/2013.

[12] B. Mitchell. (2015, 05 april 2015). Quality of Service. [Thesis]. 1.

[13] R. M. v. Selm, “TN1417 - IP QoS Standardisation for the NII (NATO Unclassified),” NATO C3 Agency, Haag08/2009.

[14] Cisco. (Januar). *DSCP and Precedence Values*. Available: http://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus1000/sw/4_0/qos/configuration/guide/nexus1000v_qos/qos_6dscp_val.pdf

[15] IETF, “Remote Authentication Dial In User Service (RADIUS),” ed: Network working group, 2000, p. 76.

[16] H. G. D. M. Andrew T. Campell, Michael E. Kounavis, Kazuho Miki, John B. Vicente og Daniel Villela, “A Survey of Programmable Networks,” Columbia University, University of Hamburg, Germany, Paper1999.

[17] O. N. Foundation, “Software-Defined Networking: The new norm for networks,” in *ONF White Paper*, ed: ONF, 2012, p. 12.

[18] A. A. C. Baker, R. Hill, N. Bessig og S.L.Kiani, “Improving cloud data center scalability, agility and performance using OpenFlow,” presented at the 2012 4th International Conference on Intelligent Networking and Collaborative Systems, Romania, 2012.

[19] O. N. Foundation, “OpenFlow Switch Specification,” ed: Open Flow Networking Foundation, 2014.

[20] T. A. N. McKeown, H. Balakrishnan, G.Parulkar, L.Peterson, J. Rexford, S. Schenker og J.Turner, “OpenFlow, enabling innovation in campus networks,” pp. 42-47, Mars, 2 2008.

[21] O. N. Foundation, “OneFlow Switch Specification,” ed: Open Networking Foundation, 2009, p. 42.

[22] M. P. Fernandez, “Evaluating OpenFlow Controller Paradigms,” presented at the ICN 2013 : The Twefth International Conference on Networks, 2013.

[23] R. R. o. J. Redi. (2012, May 2012) A Brief Overview Of Ad Hoc Networks: Challenges and Directions. *IEEE Communcations Magazine* [Invited Commentary]. 20-22.

[24] O. N. Foundation. (2012, Januar). *OpenFlow Switch Specification*. Available: https://www.opennetworking.org/

[25] K. L. H. o. D. P. Agrawal, “QoS issues in OpenFlow/SDN,” School of Computing Sciences adn Informatics, Cincinnati, 2014.

[26] N. W. Group, “RFC 1349 - Type of Service in the Internet Protocol Suite,” ed, 1992.

[27] I. 802.1, “IEEE 802.1Q,” ed: IEEE, 2012, p. 2.

[28] ONF. *Open Networking Foundation*. Available: http://www.opennetworking.org

[29] A. G. o. F. Hu, “SDN/OpenFlow: Concepts and applications,” Paper, Department of Electrical and Computer Engineering, The University of Alabama at Tuscaloosa, 2014.

[30] P. A. A. G. o. D. R. Lopez, “An OpenFlow Network Design Cycle,” Telefonica, Investigacion y Desarrollo, Madrid, Spania, 2014.

[31] N. C. F. o. L. C. S. Magalhaes, “Control and Management Software for SDNs,” Laboratorio Midiacom, Departamento de Engenharia de Telecomunicacoes, 2014.

[32] T. Z. o. F. Hu, “Controller Architecture and Performance in Software-Defined Networks,” Departmenet of Electrical and Computer Engineering, The University of Alabama at Tuscaloosa, 2014.

[33] B. A. A. N. Marc Medonca, Katia Obraczka og Thierry Turletti, “Software Defined Networking for Heterogeneous Networks,” Paper, Unitversity of California, Unitversity of California, USA og INRIA, Frankrike, IEEE MMTC E-Letters 8, , 2013.

[34] H. L. H.Yang, F. Ye, S.Lu og L. Zang, “Security in mobile ad hoc networks: challenges and solutions.,” Wireless Communications, IEEE, 2004.

[35] J. Bi, “IP Source Address Validation Solution with OpenFlow Extension and OpenRouter,” Institute for Network Sciences and Cyberspace, Tsinghua University, 2014.

[36] T. M. o. M. D. Yosr Jarraya, “A Survey and a Layered Taxonomy of Sotfware-Defined Networking,” IEEE, IEEE Communications Survey & Tutorials, Vol 16, No 4, 2014.

[37] M. F. o. F. Hu, “Language and Programming in SDN/OpenFlow,” Department of Electrical and Computer Engineering, The University of Alabama at Tuscaloosa, Tuscaloosa, Alabama, 2014.

[38] A. V. Christian Esteve Rothenberg, Marcos Rogerio Salvador, Carlos N. A Correa, Sidney Lucena, Fernando Farias, Joao Salvatti, Eduardo Cerqueira, og Antonio Abelem, “Hybrid Networking Toward a Software-Defined Era,” Faculty of Electrical and Computer Engineering, Telecomm. Research and Development Center, University of Campinas, Federal University of Para, Federal University of the State of Rio de Janeiro, Sao Paulo, Brasil, 2014.

[39] I. C. Fernando Farias, Eduardo Cerqueira, Antonio Abelem, Christian E. Rothenberg og Michael Stanton, “LegacyFlow: Bringing OpenFlow to Legacy Network Environments,” Paper, CPqD/Telecom, R&D Center, Federal University of Para, Federal Fuminense University, Brasil, Brazil, 2012.

[40] C. E. R. Marcelo R. Nascimento, Marcos R. Salvador, Carlos N. A. Correa, Sidney C. de Lucena og Mauricio F. Magalhaes, "Virtual Routers as a Service: The RouterFlow Approach leveraging Software-Defined Networks," Paper, Federal University of the Rio de Janeiro State, University of Campinas, Federal University of the Rio de Janeiro State, Brasil, 2012.

[41] P. S. Kim W, J.Lee, S.Banerjee, J.Tourrilhes, S.Lee og P.Yalagandula, "Automated and scalable QoS control for networking convergence," 2010.

[42] F. H. o. S. K. Colby Dickerson, "Multimedia over OpenFlow/SDN," Department of Electrical and Computer Engineering, Alabama at Tuscaloosa, 2014.

[43] X. F. o. F. Hu, "QoS-Oriented Design in OpenFlow," Department of Electrical and Computer Engineering, Alabama at Tuscaloosa, 2014.

[44] G. O. C. o. S. S. Sandra Scott-Hayward, "SDN Security: A Survey," Centre for Secure Information Technolocy (CSIT), Queen´s University Belfast, 2014.

[45] N. H. o. F. Hu, "Security Issues in SDN/OpenFlow," Department of Electrical and Computer Engineering, Alabama at Tuscaloosa, 2014.

[46] L. J. C. K. Benton, and C. Small, "OpenFlow Vulnerability Assessment," ACM SIGCOMM2013.

[47] W. Stallings, *Network Security Essentials: Applications and Standards*, 4th ed. Upper Saddle River: Pearson, 2011.

[48] R. E. Fairley, *The concept of operations: The bridge from operational requirements to technical specifications*, 3 ed. Colorado Springs, USA: J.C Baltzer AG, Science Publishers, 1997.

[49] FreeRadius. (2015, Desember). *The FreeRadius project*. Available: http://freeradius.org/

[50] S. V. o. S. Ganesan, “QoS Implementation For MPLS Based Wireless Networks,” Oakland University, Rochester, Michigan, 2002.

[51] GNS3. (2015, Februar). *Our Story*. Available: http://www.gns3.com/about-us.php

[52] Cisco. (2013, Cisco Packet Tracer. *8,* 1-3.

[53] nsnam.org. (2015, NS3-Network Simulator. [Veiledning]. 119. Available: https://www.nsnam.org/documentation/

[54] P. A. A. G. o. D. R. Lopez, *An OpenFlow Network Design Cycle* vol. 1. Madrid, Spain: CRC Press, 2014.

[55] D. H. Steffen Gebert, Michael Jarschel, Thomas Zinner, Phuoc Tran-Gia, “SDN Interfaces and Performance Analysis of SDN components “ 2014.

[56] E. M. Sanping Li, Yan Luo, “Programmable Network Traffic Classification with OpenFlow Extensions,” 2013.

[57] D. Erickson, “The Beacon OpenFlow Controller,” 5, Stanford University, Stanford, CA, USA, November 18, 2012.

[58] O. Alliance, “About the OSGi Service Platform,” Technical Whitepaper7 June 2007 2007.

[59] NOXRepo.org. (2014, Januar). *About NOX*. Available: http://www.noxrepo.org/nox/about-nox/

[60] NOXRepo.org. (2014, Januar). *About POX*. Available: http://www.noxrepo.org/pox/about-pox/

[61] Floodlight. (November 18, 2012, November). *Floodlight is an Open SDN Controller*. Available: http://www.floodlight.openflowhub.org/

[62] opendaylight.org. (2015, Oktober). *About Open Daylight*. Available: http://www.opendaylight.org/project/about

[63] Kanika. (2014, Januar). *OpenDaylight & OSGI* [Veiledning]. Available: http://sdntutorials.com/opendaylight-and-osgi/

[64] Cisco. (2015, Oktober). *Cisco´s One Platform Kit (onePK)*. Available: http://www.cisco.com/c/en/us/products/ios-nx-os-software/onepk.html

[65] Trema. (2015, Oktober). *Trema - Full-Stack OpenFlow Framwork*. Available: http://trema.github.io/trema/

[66] Cisco, "Cisco Open Network Environment: Bring the Network Closer to Applications," April 2014.

[67] S. Microsystems, "RFC 5531 - Remote Procedure Call Protocol Specification," ed: Network Working Group, 2009, p. 62.

[68] Cisco. (2015, Desember). *Base Service Set*. Available: https://developer.cisco.com/media/onePKJavaAPI-v1-1-0/index.html

[69] LinuxContainers.org. (2015, Februar). *LinuxContainers.org Infrastructure for container projects*. Available: https://linuxcontainers.org/#

[70] Q.-o. s. p. emulator. (Januar). *KVM*. Available: http://wiki.qemu.org/KVM

[71] C. N. Academy. (2015). *Cisco Packet Tracer*. Available: https://www.netacad.com/web/about-us/cisco-packet-tracer

[72] Cisco. (2015, November). *Cisco DevNet*. Available: https://developer.cisco.com/site/devnet/home/index.gsp

[73] SLF4J. (2014, November). *Simple Logging Facade for Java*. Available: http://www.slf4j.org/

[74] T. O. W. A. S. Project. (2015, Januar). *Certificate and Public Key Pinning*. Available: https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning

[75] Cisco, "Tech Note: onePK TLS Certificate Pinning and TLS Debugging," May 30 2014.

[76] T. Taubert. (2103, Desember). *Deploying TLS the hard way*. Available: https://timtaubert.de/blog/2014/10/deploying-tls-the-hard-way/

[77] Oracle. (2013, Januar). *Java Security Overvierw*. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/security/overview/jsoverview.html

[78] J. d. Lavarene, "SSL with Oracle JDBC Thin Driver," ed: Oracle, 2010.

[79] CISCO. (2014). *Configuring RADIUS*. Available: http://www.cisco.com/c/en/us/td/docs/ios/12_2/security/configuration/guide/fsecur_c/scfrad.html

[80] Cisco. (2015, November). *Cisco onePK Java Reference*. Available: https://developer.cisco.com/media/onePKJavaAPI-v1-1-0/index.html

[81] Cisco. (2013, Januar). *Access Control Lists: Overview and Guidelines*. Available: http://www.cisco.com/c/en/us/td/docs/ios/12_2/security/configuration/guide/fsecur_c/scfacls.html

[82] IETF, "RFC 3550," in *RTP*, ed: Network Working Group, 2003.

[83] IETF, "RFC 5865," in *A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic*, ed: Network 2010.

[84] J. Community. (2012, Januar). *Traffic Engineering 2: Layer 2 Prioritisation - CoS (Class of Service)*. Available:

https://community.ja.net/library/videoconferencing-booking-service/traffic-engineering-2-layer-2-prioritisation-cos-class-0

[85] IETF, "RFC 5246," in *The Transport Layer Security (TLS) Protocol*, ed: Network Working Group, 2008.

[86] K. C. J.Babiarz, "RFC 4594 - Configuration Guideline for DiffServ Service Class " pp. 9-10, 2006.

[87] Forsvarsdepartementet, "Beskrivelse av programområde informasjonsinfrastruktur," Forsvarsdepartementet2005.

[88] Forsvarsdepartementet, "Policy for militær tilpasning og anvendelse av informasjons- og kommunikasjonsteknologi i Forsvaret," vol. 12, 2005.

[89] P. 9271, "Vedlegg A - Kravdokument Taktiske Kommunikasjonsløsninger," Forsvarsdepartementet, Ed., 2.0 ed: Forsvaret, 2013, p. 132.

[90] F. Sørensen, *Moderne IP-nett*, 2 ed.: Bookworld, 2008.

[91] IETF, "RFC 1633," in *Integrated Services in the Internet Achitecture: An Overview*, ed: Networking Working Group, 1994.

[92] IETF, "RFC 2475," in *An Architecture for Differentiated Services*, ed: Network Working Group, 1998.

[93] IETF, "RFC 2748 COPS protocol," ed, 2000.

[94] E. A. Mankin, F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, L. Zhang, "Resource ReSerVation Protocol (RSVP)," ed: Network Working Group, 1997.

[95] Cisco, "Guide to ATM technology," vol. 262, 2010.

[96] IETF, “RFC 3439,” in *Some Internet Architectural Guidelines and Philosophy*, ed: Network Working Group, 2002.

[97] IETF, “RFC 5036,” in *LDP Specification*, ed: Network Working Group, 2007.

Vedlegg A
Forsvarets kommunikasjonsinfrastruktur (FKI)

Kommunikasjonstjenester

Forsvaret har i sin materiellplan[87] og IKT-policy[88] vedtatt å innføre en felles kommunikasjonsinfrastruktur for alle kommunikasjonstjenester. En kommunikasjonstjeneste kan defineres som overføring av informasjon med et gitt sett av egenskaper. Egenskapene er i hovedsak sikkerhetsmessige eller funksjonelle. Tabell AT1 under viser en oversikt over noen kommunikasjonstjenester:

Kommunikasjonstjeneste	Egenskap	Beskrivelse
Sanntid og lav hastighet	Funksjonell	Telefoni og sensorsystemer
Sanntid og høy hastighet	Funksjonell	Videokonferanse og bildedannende sensorer
Interaktive tjenester	Funksjonell	K2-systemer, administrative systemer, web løsninger
Meldingsbasert datautveksling	Funksjonell	Meldingstjenesten, filoverføring, K2-systemer
Konfidensialitet	Sikkerhet	Kryptosystemer
Tilgjengelighet	Sikkerhet	Plattform, aksesskontroll, autentisering
Integritet	Sikkerhet	Kryptosystemer

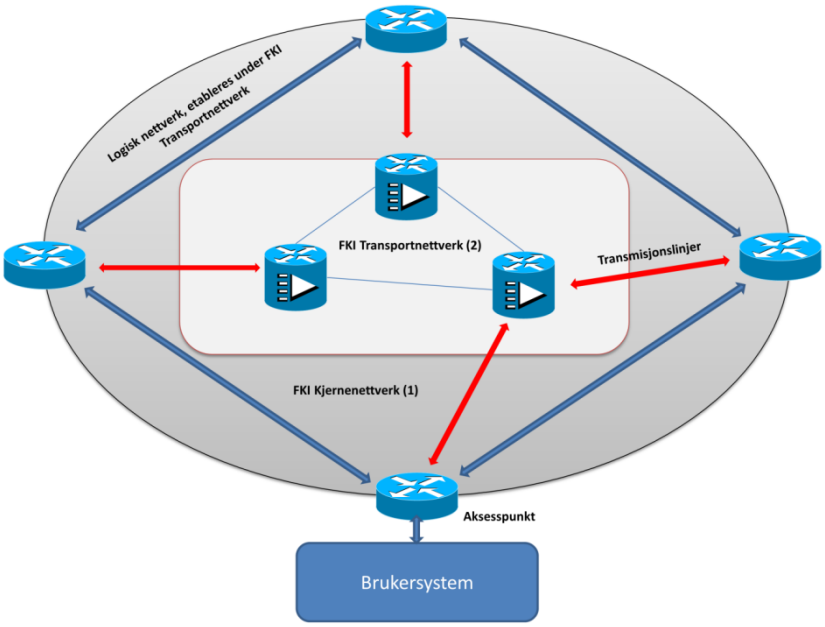
AT 1: Kommunikasjonstjenester

Noen tjenester dekkes i FKI, noen delvis men flere sikkerhetstjenester støttes per dags dato ikke. De fleste sikkerhetsegenskapene er stort sett dekket av brukersystemene.
Forsvaret har begynt å innføre en del applikasjoner som realiseres gjennom bruk av et antall felles kjerne- og funksjonsvise tjenester, målsetningen på sikt er én felles plattform som produserer alle typer kommunikasjonstjenester, dette vil gi økt interoperabilitet og større fleksibilitet. I tillegg vil dette forenkle drift og kunne redusere kostnader. Men opprettelse av et datasenter og innføring av nye tjenester i FKI vil medføre til at nye tjenester i seg selv gir økt trafikk og at kompleksiteten blir større. Dette er viktige hensyn som må vurderes i forhold til valg og utvikling av nettverksarkitekturen.

Elementer i FKI

FKI består av flere elementer. De fleste elementene finner man igjen i en generell beskrivelse av kommunikasjonsinfrastruktur, men på grunn av Forsvarets sikkerhetsløsning og kravet til sikkerhet i nye implementeringer og arv, er arkitekturen noe mer komplisert og nyansert.
Noen av de fysiske egenskapene en kommunikasjonsinfrastruktur kan ha er som følger:

- En installert base som er konstant i utvikling.
- Åpen, som betyr at brukerne selv kan benytte ulike teknologier, løsninger eller systemer for å bruke infrastrukturen.
- Den kan bestå av mange forskjellige teknologier, heterogen.
- Den er delt mellom brukerne. Den kan benyttes av alle brukerne, ingen har eierrettigheter.



AF 1: FKI Kjernenettverk

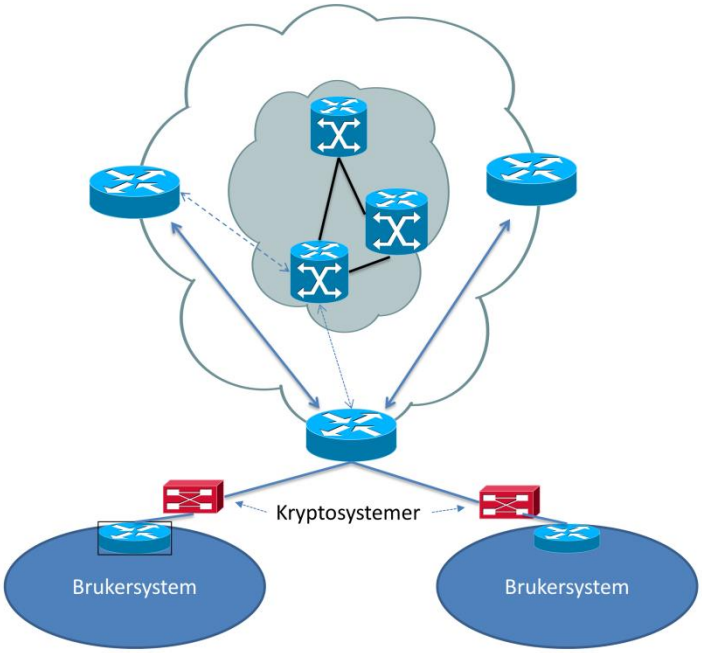
FKI Kjernenettverk(1) figur AF1, produserer kommunikasjonstjenester til brukerne. Forvaltning av ressurser, prioriteringer og kapasitetstildeling til brukersystemene skjer i kjernenettverket. Kjernenettverket har strenge krav til oppetid[25] og tilgjengelighet. Kommunikasjon fra brukerens infrastruktur og inn til kjernenettverket skjer gjennom definerte grensesnitt kalt aksesspunkt.

FKI Transportnettverk(2) figur AF1, kan være radiolinjesystemer, egne og kontrollerte fiber kapasiteter og innleide ressurser. Deployerbare systemer som radiosystemer og SATCOM er også en del av det. Kjernenettverket bruker FKI Transportnettverket og kommunikasjonstjenestene er avhengig av den fordi den binder nodene sammen. Kvalitet på forbindelsene, fysisk lokasjon og sårbarheter er eksempler på egenskaper for transportnettverket som det er viktig for Forsvaret å ha oversikt over. Dette har også en sammenheng med ressursstyringen og overordnet drift.
Blå pil i figur AF1 indikerer det logiske nettverket i kjernenettverket som etableres under FKI transportnettverket. Rød pil er kommunikasjon i transportnettverket som binder nodene sammen.

[25] Oppetid beregnes som den totale tiden løsningen eller tjenesten fungerer i forhold til kundens krav

For å få aksess til kommunikasjonstjenestene kan **brukersystemene** (AF 1) benytte flere ulike teknologier, eksempelvis radio, fiber, ICE eller Ethernet. Et stort spekter fra sivile til militære eller NATO standarder. Interoperabilitet mellom ulike teknologier og grensesnittene blir dermed viktig for å få aksess til kommunikasjonstjenestene i FKI.

Et brukersystem er noe eller noen som benytter seg av kommunikasjonstjenestene. Det kan være en bruker, en terminal, en sensor eller en applikasjon. Mange av dagens brukere er store nettverk i seg selv og benytter seg av den felles infrastrukturen som et transportnettverk for sin egen infrastruktur. Brukersystemene kan ha ulike kommunikasjonsbehov og kan derfor ikke bruke samme løsning eller teknologi. Det må derfor være en løs kobling mellom brukersystemer og transportnettverket.



AF 2: Produksjonstjenester for kommunikasjonstjenester

FKI kjernenettverket i AF1 må kunne tilby kommunikasjonstjenestene til mange ulike brukersystemer og plattformer. Kjernenettverket trenger avanserte kommunikasjonstjenester for å kunne tilby differensierte og kvalitetssikrede tjenester til alle de forskjellige systemer som finnes i Forsvaret. Figur AF2 viser plasseringen kryptosystemer som setter restriksjoner på kommunikasjonen mellom nettverket i brukersystemene og kjernenettverket, Forsvaret trenger derfor enheter som kan levere kommunikasjonstjenester også internt i brukersystemene ettersom applikasjoner og tjenester på plattformen må ha tilganger til det.
Kryptosystemene skaper store utfordringer når det gjelder mulighet for ressurshåndtering og trafikkstyring. Forsvaret tillater noe informasjon å slippe gjennom kryptoapparatet, men

dette er ikke nok til for eksempel få til prioritering av operativ viktighet mellom applikasjoner på tvers av forskjellige domener.

TOS

Taktiske områdedekkende sambandssystem (TOS) levert gjennom et prosjekt i Forsvaret, er å anse som et element i den taktiske delen av FKI. Forbindelse med FKI skjer gjennom definerte termineringspunkter eller gjennom satellittkommunikasjon og vil sørge for at det er mulig og tilgjengeliggjøring av tjenester fra den strategiske del av FKI ned til kommandoplasser eller enheter i det taktiske domenet. Prosjektet har flere formål [89]:

- Øke kommunikasjonssystemets kapasitet og fleksibilitet.
- Innføre et kommunikasjons lag basert på IP-teknologi.
- Etablere management for styring og overvåking.
- Bidra til økt mobilitet i operasjoner.
- Gir mulighet for interoperabilitet ved å innføre nye grensesnitt.

Prosjektet har avhengigheter til andre viktige kommunikasjonsprosjekter i Forsvaret, eksempelvis nye radioer med mer båndbredde kapasitet og standardisering av grensesnitt for interoperabilitet mellom ulike nettverk for å nevne noen.

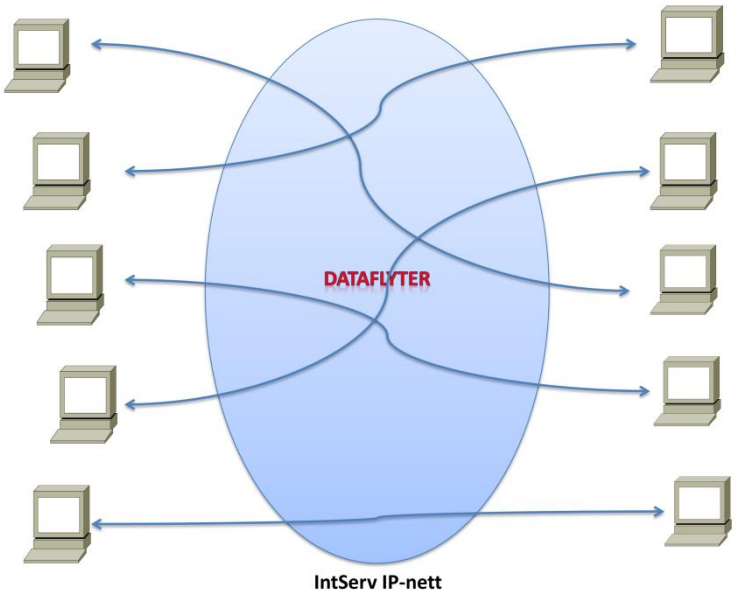
Et dimensjonerende krav til brukeren av systemet vil være evne til å gjennomføre fullspektrum høyintensitets samvirkeoperasjon. En slik operasjon stiller store krav til blant annet mobilitet og tempo. Det blir tilgjengeliggjort flere type sensorer, effektorer og beslutningstakere som medfører til at datamengden øker. Situasjonsbevissthet i et raskt skiftende dynamisk operasjonsområde er man avhengig av nær sanntids informasjon. Konnektivitet i nettverk og utveksling av informasjon bidrar til en situasjonsforståelse som er nødvendig.

Vedlegg B
Tjenestekvalitet i IP nettverk

IntServ

Integrated Services (IntServ) er en arkitektur for å støtte alle typer tjenester i ett felles IP-nett. Arkitekturen bygger på at IP-trafikken består av et stort antall dataflyter (figur BF1), og ved å håndtere disse flytene ut fra deres individuelle behov, kan man sikre at nødvendige ressurser er tilgjengelige til enhver tid. Applikasjonene eller brukerne må angi parametere for dataoverføringens båndbredde og eventuelt andre krav. Nettverket må deretter forsikre seg om at disse parameterne kan tilfredsstilles uten å gå på bekostning av andre overførings kvalitetskrav. IntServ-rutere må kunne håndtere tilstandsinformasjonen for de ulike dataflytene. Arkitekturen innfører også en egen ressursreserveringsprotokoll som sørger for oppkobling av dataflytene som ikke må forveksles med forbindelser. IP-nettet skal fortsatt fungere som et forbindelsesløst nettverk. *En dataflyt er en strøm av sammenhørende pakker som er resultat av en enkelt brukeres aktivitet og krever en bestemt tjenestekvalitet [90]*.

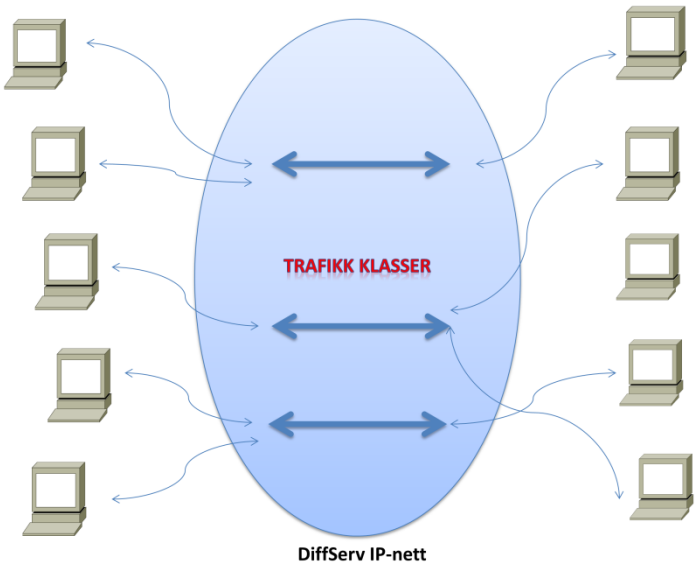
IntServ-arkitekturen er arbeidsintensiv for ruterne, spesielt i et kjernenettverk der det er mange dataflyter. IntServ må dessuten bli konfigurert på alle enheter/rutere langs ruten til dataflyten. På grunn av dette skalerer ikke IntServ godt i store nettverk [91]. Dessuten er den harde reservasjonen av båndbredde ofte bortkastet og kan medføre til dårlig utnyttelse av båndbredden for annen type trafikk.



BF 1: IntServ

DiffServ

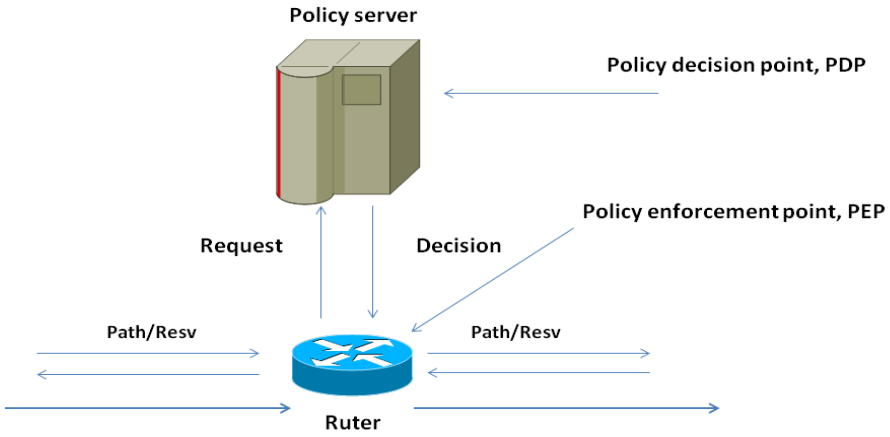
Differentiated Services (DiffServ) [92] er en mekanisme som tildeler tjenestediskriminering i nettverket. Funksjonaliteten til ruterne i kjernenettverket er relativ enkel, mens de mer kompliserte ruterne på kanten av nettverket tar seg av å plukke ut dataflyter og sette et prioriteringsnivå på de ulike dataflytene. Tjenestediskriminering er implementert på rutere uten å koordinere med andre rutere langs ruten. Inni kjernenettverket blir ikke pakkene håndtert per dataflyt men per prioriteringsnivå kalt trafikklasser (figur BF2). DiffServ jobber på en forestilling om at IP pakker uttrykker deres krav til nettverket for eksempel ved hjelp av kode bit (Differentiated Services Code Points, DSCP) i IP hodet, og at hver ruter i hvert hopp prøver å tildele kravene etter best mulig evne (Per Hop Behaviour, PHB). DiffServ gir ingen harde garantier eller tilbakemelding til applikasjonen eller tjenesten at forespurte differensierte tjenester vil bli møtt. Prosesseringsbelastningen til ruterne i en DiffServ arkitektur blir mye mindre enn ved IntServ og sikrer bedre skalering av tjenestekvalitetsarkitekturen.



BF 2: DiffServ

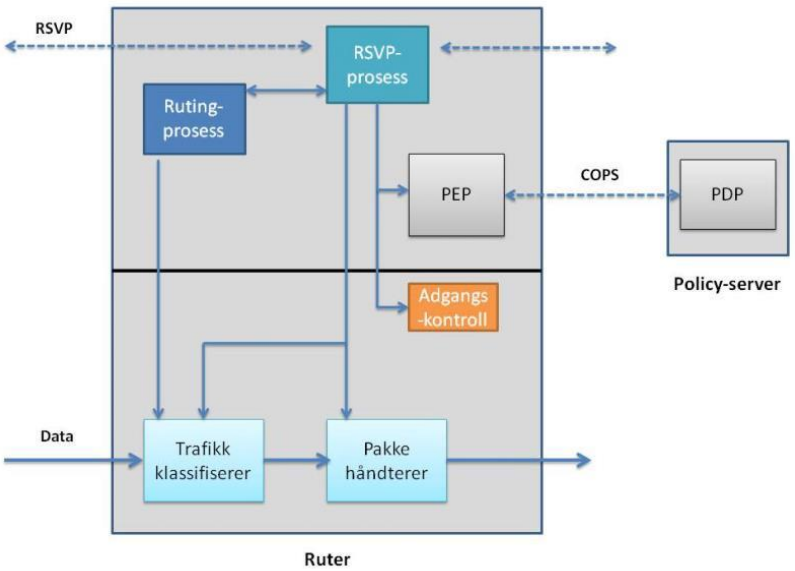
Policykontroll

Mekanismer for tjenestekvalitet i et nettverk betyr at datatrafikk fra applikasjoner eller tjenester kan få bedre ytelse på bekostning av andre. En mekanisme som kontrollerer og begrenser hvem som får tilgang til god tjenestekvalitet, i tillegg til å bestemme hvilke applikasjoner det skal gjelde for og når og hvor lenge denne tilgangen skal gis, kalles policykontroll. IETF (Internet Engineering Task Force) har utarbeidet RFC2748 [93], et rammeverk for hvordan policykontroll kan implementeres i et IP-nettverk. Den består av: *Policy Decision Point* (PDP), policyavgjørende punkt og *Policy Enforcement Point* (PEP), policyutøvende punkt (figur BF3).



BF 3: Policykontroll

PEP er en del av selve nettverket og har ansvaret for å begrense tilgangen til ressursene ut fra avgjørelser som tas i PDP. PDP kan være plassert i en ekstern policyserver og bruker policyregler for å ta avgjørelser som formidles til PEP, eller anvende eksempelvis en ekstern autentiseringsserver. Figur BF4 illustrerer et eksempel på anvendelse i en IntServ ruter med bruk av *Resource Reservation Protocol* (RSVP) [94] protokollen, der reservasjon av ressurser avgjøres av PDP. PEP eksisterer som regel kun i ruterne på kanten av domenet og sørger for å ha kontrollen i nettverket.



BF 4: Policykontroll og RSVP. Kommunikasjon mellom PEP og PDP ved hjelp av COPS[26]

[26] Common Open Policy Service (COPS) er definert for kommunikasjon mellom PEP og PDP. Meldingsutveksling mellom PEP og PDP består av Request/Decision-meldinger.

Vedlegg C
Noen nettverksteknologier og sitt forhold til tjenestekvalitet

Best-effort nettverk

IP nettverk er beskrevet ofte som *best-effort* nettverk. Dette refereres til et nettverk som ikke aktivt differensierer behandlingen av trafikken fra ulike tjenester i nettverket. I et *best-effort* nettverk blir alle IP pakkene behandlet likt. Nettverket foretar sin *best-effort* for å levere hver pakke så raskt som mulig, men gjør ingen forpliktelser til å behandle enhver klasse av pakker fortrinnsvis til noe annet. Selv om dette høres ut som en helt upartisk og rettferdig tilnærming, sies det at Internett-nettverk som for visse applikasjoner og tjenester er *best-effort* rett og slett ikke er god nok. Det hevdes at det IP-nettverk trenger er en eller annen metode for å gi en overlegen respons for å støtte visse klasser av programmer på noen spesiell måte. Bedre enn *best-effort* er en form for å beskrive tjenestekvalitet på.

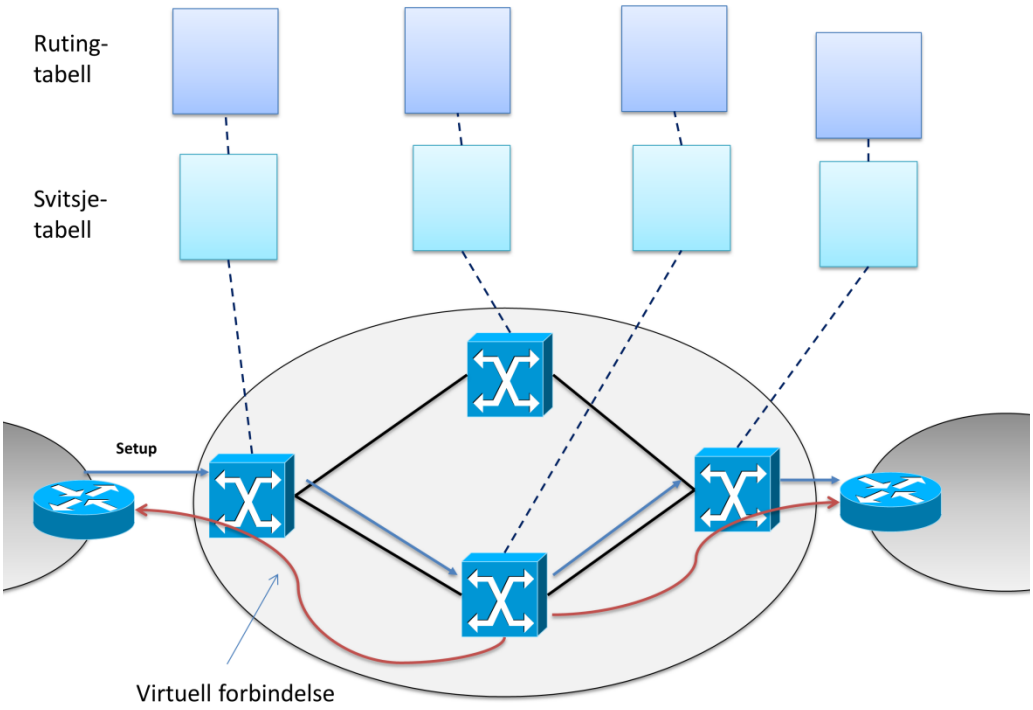
ATM

Asynchronous Transfer Mode (ATM) er en forbindelsesorientert teknologi som ble utviklet tidlig på 90-tallet av teleorganisasjonene med tanke på det kommende offentlige tjenesteintegrerte bredbåndsnettet med overføringshastigheter opp til 155 Mbit/s og høyere. I et ATM-nettverk transporteres informasjonen i små datapakker som kalles ATM-celler. Hver celle er 53 byte lang og kan frakte 48 byte informasjon. For å frakte store datamengder mellom to terminaler overføres mange ATM-celler i nettverket. Cellene som følger den samme veien gjennom nettet kalles en *virtuell forbindelse*. Forbindelsene kan være permanent Permanent Virtual Connection (PVC) eller ved behov Switched Virtual Connections (SVC).

ATM-svitsjene er utstyrt med en ruting tabell og en svitsje tabell (figur CF1). S*etup*-meldingene rutes via ruting tabellen. Når analysen foretas tilordnes den virtuelle forbindelsen en indikatorverdi, link-lokal indikator, som identifiserer forbindelsen på linken. Indikatorverdiene legges inn i svitsjetabellene på hver link den passerer mellom de to endepunktene som utgjør den virtuelle forbindelsen. Når *setup* er utført merkes dataen med aksesspunktets link-lokale indikator, som blir byttet ut med den som gjelder for neste link i hver svitsj som passeres gjennom hele forbindelsen. Dataen sendes på denne måten hele veien frem til den andre enden. Svitsjingen utføres typisk i maskinvaren og blir dermed mer effektiv enn ruting av *setup*-meldingen som dessuten har flere adresser i ruting tabellene enn det er link-lokale indikatorer i svitsjetabellen. Størrelsen på de link-lokale indikatorene

er dessuten fast, mens størrelsen på adresseprefiksene varierer.

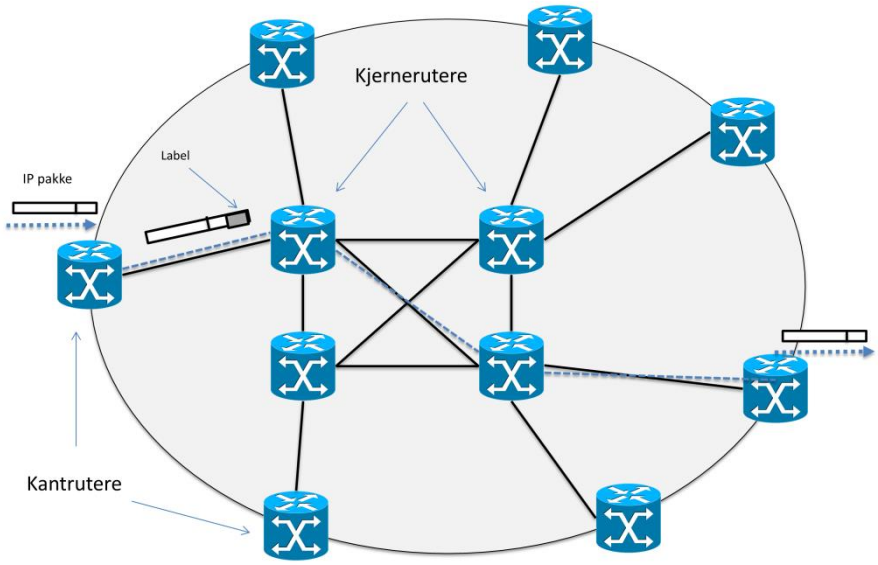
ATM-cellene er som beskrevet over veldig små, IP-pakkene passer normalt sett ikke inn i cellene. En IP-transport over en virtuell forbindelse utstyres med et tilpasningslag AAL (ATM Adaptation Layer)[95] i begge ender av forbindelsen som deler IP-pakkene opp i mindre segmenter når dataen sendes over en virtuell forbindelse. AAL reassemblerer cellene igjen i andre enden av den virtuelle forbindelsen.



CF 1: ATM-nettverk

MPLS

Multi-Protocol Label Switching (MPLS) er en standard som er utformet generelt for alle typer lag3-protokoller [96], en multi-protokoll som kombinerer fleksibiliteten til IP-rutingen med effektiviteten til svitsjingen. [27]



CF 2: MPLS Domenet

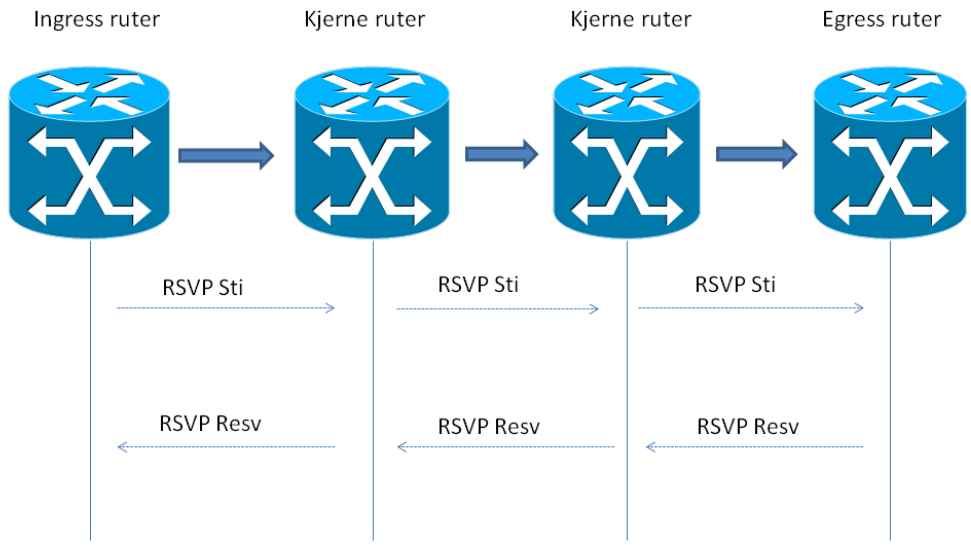
Ruting og videresending er to hovedfunksjoner i et MPLS nettverk. Ruting utføres på lik linje som i tradisjonelle IP nettverk, men videresending utføres på laget under. Dette kan man få til ved hjelp av en link-lokal identifikator, en merkelapp (*label*) som settes foran IP-pakken. Denne merkelappen utfører videresendingen på lik linje som ATM-teknologien. For å koble lag 3-rutingen med lag 2-svitsjingen må merkelappene distribueres mellom ruterne/svitsjene ved hjelp av en distribusjonsprotokoll.
Et MPLS domenet (CF2) består av et sett med rutere som kalles *Label Switching Routers* (LSR). Disse kan ha ulike roller og gjennomfører ulike oppgaver basert på hvor de er plassert i nettverket. Kantrutere knytter MPLS-domenet til andre noder på utsiden, de andre er definert som kjernerutere. En IP-pakke som skal sendes gjennom MPLS-domenet ankommer først ingresskantruteren som vil bruke IP-adressen fra hodet og slå opp i ruting tabellen. Innslagene i ruting tabellen er også utstyrt med en såkalt *Next Hop Label Forwarding Entry* som inneholder merkelapp-verdien som svarer til denne ruten. Ingresskantruteren vil da legge til merkelappen til IP-pakken og deretter videresende pakken til neste ruter.

[27] Lag 2, linklaget, OSI modellen

Hver kjerneruter som pakken passerer på veien vil videresende pakken på bakgrunn av merkelappen og ikke IP-adressen i IP-hodet. En kjerneruter inneholder en svitsjetabell som den slår opp i og bytter ut innkommende merkelapp-verdi med en utgående verdi. Dette er en prosess som kan gjøres i hardwaren og veien gjennom domenet kalles en Label Switched Path (LSP).
Når IP-pakken kommer frem til egress kantruteren vil den ta bort merkelappen og slå opp i ruting tabellen ved hjelp av IP-adressen i IP hodet. Distribusjonen av merkelapper (label) i et MPLS domenet dekkes ikke i dette vedlegget, mer informasjon finnes her: [97].

RSVP og DiffServ over MPLS

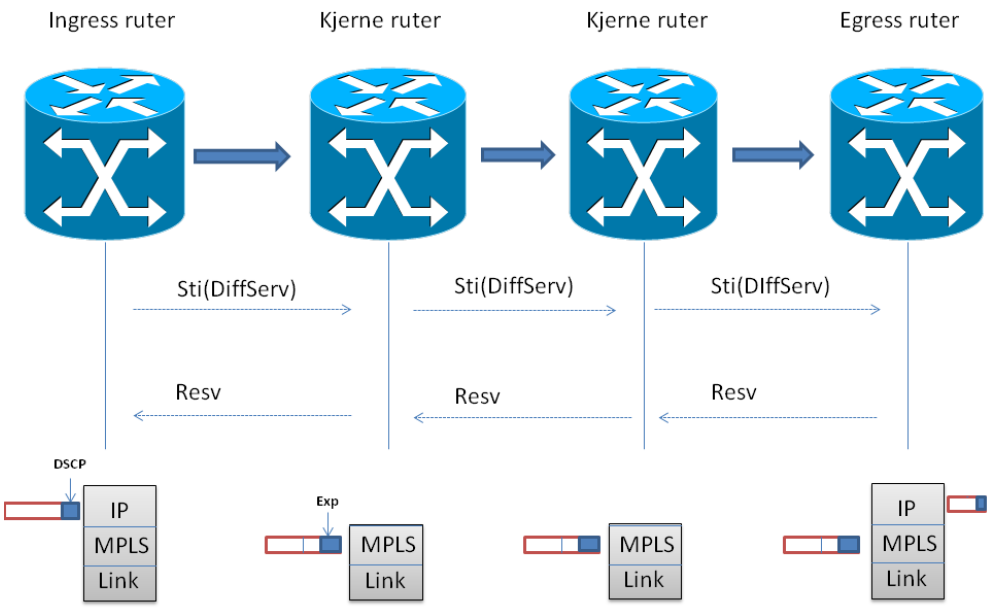
Resource Reservation Protocol (RSVP) [94] er utviklet for å utføre ressursreservering over IP-nettverk, en utvidelse er spesifisert for denne protokollen i MPLS slik at man kan utføre videresending med tjenestekvalitet i domenet. Utvidelsen gjør det mulig og eksplisitt styre oppkoblingen av LSP-stier.



CF 3: MPLS og RSVP

Skissen over (CF3) viser hvordan en LSP-sti settes opp med RSVP. LSP stiene er relativt statiske men kan settes opp til å re-rutes ved behov for å oppnå best mulig effektiv kapasitetsutnyttelse og optimal ytelse av nettverket.
Styring av trafikk gjennom et nettverk langs en forutbestemt stil kalles *Traffic Engineering* (TE). Med bruk av vanlig IP-ruting blir kontroll over ressursbruken i et nettverk umulig. Forbindelsesorienterte teknologier som ATM som setter opp faste stier, gjør det enklere å

sørge for tjenestekvalitet i nettet. TE er en måte å oppnå en liknende funksjonalitet på i et MPLS-basert IP-nett.



CF 4: Signalering, DiffServ over MPLS

MPLS-teknologi kan også implementere DiffServ-arkitekturen (figur CF4) ved å tilordne LSP-stier til de ulike tjenesteklassene. Ved hjelp av Exp-feltet i MPLS hodet (figur CF4) kan det spesifisere to typer LSP-stier: 1) Exp-indikert, da vil en LSP-sti kunne støtte opp til 8 ulike tjenesteklasser. Exp-feltet brukes til å indikere hvilken PHP som skal anvendes på en enkelte pakke. 2) Merkelapp indikert, da vil det opprettes en sti per PHB-gruppe.

Selve den konkrete tilknytningen mellom kodingen i Exp-feltet og DiffServ-tjenesteklasser kan avgjøres enten ved konfigurasjon på forhånd i MPLS-ruterne, eller signaleres i forbindelse med oppkoblingen av den enkelte LSP-sti.
MPLS kan håndtere tjenestekvaliteten på samme måte som IP-laget men benytter seg av merkelappen og Exp-feltet for å peke ut tjenesteklassen.

Vedlegg D
Nettverkstopologi i vmcloud

Under er et eksempel på XML script for å lage en nettverkstopologi i vmcloud:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<topology xmlns="http://www" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemaVersion="0.3"
xsi:schemaLocation="http://www.localhost/virl.xsd">

  <node name="router1" type="SIMPLE" subtype="vios" location="188,263" vmImage="/usr/share/vmcloud/data/images/vios.ova">

    <extensions>

      <entry key="bootstrap configuration" type="String">/home/3node/router1.con</entry>

      <entry key="import files" type="String">/home/3node/router1.p12</entry>

    </extensions>

    <interface name="GigabitEthernet0/0"/>

    <interface name="GigabitEthernet0/1"/>

    <interface name="GigabitEthernet0/2"/>

    <interface name="GigabitEthernet0/3"/>

  </node>

  <node name="router2" type="SIMPLE" subtype="vios" location="488,319" vmImage="/usr/share/vmcloud/data/images/vios.ova">

    <extensions>

      <entry key="bootstrap configuration" type="String">/home/3node/router2.con</entry>

      <entry key="import files" type="String">/home/3node/router2.p12</entry>

    </extensions>

    <interface name="GigabitEthernet0/0"/>

    <interface name="GigabitEthernet0/1"/>

    <interface name="GigabitEthernet0/2"/>

  </node>

  <node name="lan_ex" type="SEGMENT" location="722,161"/>

  <connection src="/topology/node[1]/interface[1]" dst="/topology/node[2]/interface[1]"/>

  <connection src="/topology/node[1]/interface[2]" dst="/topology/node[3]/interface[1]"/>

  <connection src="/topology/node[1]/interface[3]" dst="/topology/node[4]"/>

  <connection src="/topology/node[2]/interface[2]" dst="/topology/node[4]"/>

  <connection src="/topology/node[3]/interface[2]" dst="/topology/node[4]"/>

  <connection src="/topology/node[3]/interface[3]" dst="/topology/node[6]"/>

46 av 250 kodelinjer
```

DF 1: XML script - nettverkstopologi

Vedlegg E
Nettverkstopologi – utdrag fra diagnostikk

Under viser en utskrift fra diagnostikk av nettverkstopologien.

cisco@onepk:~$ vmcloud netdiag -v 3node

Using default configuration: /etc/vmcloud/vmcloudrc

List VMs and connections in topology '3node' ...

Node: router1

state max mem(kb) current mem(kb) num_of_cpu cpu_time(sec)

running 393216 393216 1 64079

Interfaces:

Name Mac_Address Tap Interface Routing Device

GigabitEthernet0/0 52:54:00:54:71:24 vt8cc892e6aa48b vb702c316214354

GigabitEthernet0/1 52:54:00:e4:5d:08 vt2e3c71d48e6ed vbaa1035afb78ef

GigabitEthernet0/2 52:54:00:ce:06:61 vtf8d0b10d8011f vb_vmc_lan_1

GigabitEthernet0/3 52:54:00:8c:88:db vt31fe0fd462c53 vb_ex_eth1

Console: telnet 127.0.0.1 3534

Aux : telnet 127.0.0.1 3536

Node: router2

state max mem(kb) current mem(kb) num_of_cpu cpu_time(sec)

running 393216 393216 1 64056

Interfaces:

Name Mac_Address Tap Interface Routing Device

GigabitEthernet0/0 52:54:00:62:31:f7 vt13e7bfb584c17 vb702c316214354

(..)

Vedlegg F
Scenario: Tilgang til kjernetjenester for TKN
Logger og utskrifter fra tester

Utdrag fra utskrift fra konsoll i Eclipse, kjøring av programmet *AServivce*. Sesjonsnummer vises helt til venstre. Logginfo er programmert i koden slik at det blir enklere å forstå hva som skjer.

Test 1:

Oppsett av TLS.

Under viser en dialogboks for autentisering av sertifikat for bruker (FF 1):

FF 1: Sertifikat autentisering - oppsett av TLS

```
187  [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Kommunikasjon ved hjelp av -
TLS
177132 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Forbindelse etablert -
NetworkElement [ 192.168.40.144 ]
       Product ID   : IOSv
       Processor    : IOSv Chassis
       Serial No    : 98ARNL3COP87CK3FT586Q
       Version ID   : null
       sysName      : Router2
       sysUpTime    : Wed Dec 31 16:03:20 PST 1969
       sysDescr     : Cisco IOS Software, IOSv Software (VIOS-ADVENTERPRISEK9-M),
Experimental Version 15.4(20140730:011659) [lucylee-pi25-2 107]
Copyright © 1986-2014 by Cisco Systems, Inc.
Compiled Tue 29-Jul-14 18:17 by lucylee
sysObjId     : 1.3.6.1.4.1.9.1.1041
177133 [main] INFO  com.cisco.onep.tkn.InterfaceConN  -
******* Autentiserer *******
(…)
```

Test 2:

Test av *AService* for bruker *uone:*

```
177518 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Autentisering OK for: uone
177519 [main] INFO  com.cisco.onep.tkn.InterfaceConN  -
******* Henter AAA Server info *******
177519 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Server IP addresse:
/192.168.0.106
Protocol: ONEP_AAA_PROTOCOL_RADIUS
177520 [main] INFO  com.cisco.onep.tkn.InterfaceConN  -
******* Henter tilgangsprofil *******
177520 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Auto-accounting is enabled.
177520 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Velg metode: [ list (for å
liste konfig) | endre | rediger ]
177520 [main] INFO  com.cisco.onep.tkn.InterfaceConN  -
Velg metode eller skriv avslutt for å avslutte:
endre
387013 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Du å se konfig på
interfacene:
391275 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - CLI Result (show run) -
Building configuration...
```

(…)

Test 3:

Test av *AService* for bruker *cops:*

```
6770 [main] INFO  com.cisco.onep.tkn.InterfaceConN  -
******* Autentiserer *******
6964 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Autentisering OK for: cops
6966 [main] INFO  com.cisco.onep.tkn.InterfaceConN  -
******* Henter AAA Server info *******
6966 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Server IP addresse:
/192.168.0.106
Protocol: ONEP_AAA_PROTOCOL_RADIUS
6966 [main] INFO  com.cisco.onep.tkn.InterfaceConN  -
******* Henter tilgangsprofil *******
6966 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Velg metode: [ list (for å
liste konfig) | endre | rediger ]
6966 [main] INFO  com.cisco.onep.tkn.InterfaceConN  -
Velg metode eller skriv avslutt for å avslutte:
endre
51999 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Du valgte å se kjørende
konfig på ruter:
55000 [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Ikke tillatt valg.
55000 [main] INFO  com.cisco.onep.tkn.InterfaceConN  -
Velg metode eller skriv avslutt for å avslutte:

(..)
```

Test 4:

Kjøring av *AService* for *testuser*

```
Most recent failed connection attempts:

Connection #1 attempted Thu Feb 19 19:19:31 2015
Remote host: 192.168.40.132
Reason: Authentication failed for user INVALID application
com.cisco.onep.tkn.AService-onepk
Reason code: 2
Connection sequence number: 51
```

Kommentar: Bruker *testuser* har ikke tillatelse til å kjøre applikasjonen på grensesnittet.

Test 5:

Brudd i kommunikasjonskanalen mellom brukerapplikasjon og nettverkselementet, kontroller returnerer følgende meldinger:

```
0    [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Forbindelse med følgende
nettverksenhet: -
NetworkElement [ router2]
11   [main] INFO  com.cisco.onep.tkn.InterfaceConN  - Kommunikasjon ved hjelp av -
TLS

(…)

63565 [main] ERROR com.cisco.onep.element.NetworkElement  - Transport exception:
org.apache.thrift.transport.TTransportException: Could not connect to router2 on
port 15002
```

```
63568 [main] WARN  com.cisco.onep.element.NetworkElement  - Failed to connect to
router2/10.10.10.120: Error occurred in the operation. Failed to connect to the
network element or the session is closed. Could not connect to router2 on port
15002

63568 [main] ERROR com.cisco.onep.tkn.InterfaceConN  - Error occurred in the
operation. Failed to connect to the network element or the session is closed. Could
not connect to router2 on port 15002
```

Prosess kjører ennå på ruter, men keepalive er satt til 60 sekunder.

```
Router2#show onep statistics session all
Session ID: 3389
Application Name: ASerivce
API In: 14          API Out: 5
Bytes In: 3658            Bytes Out: 10669
Vty Count: 0
Memory Allocated: 917400 bytes   Memory Freed: 911088      Memory Held: 31448
CPU utilization for five seconds:  0.0 %      one minute:  0.0 %  five minutes:  0.0
%


574347 [pool-1-thread-1] ERROR com.cisco.onep.core.event.KeepaliveMonitor  - no
keepalive event from network element!
```

Timeout etter 60 sekunder

Test 6:

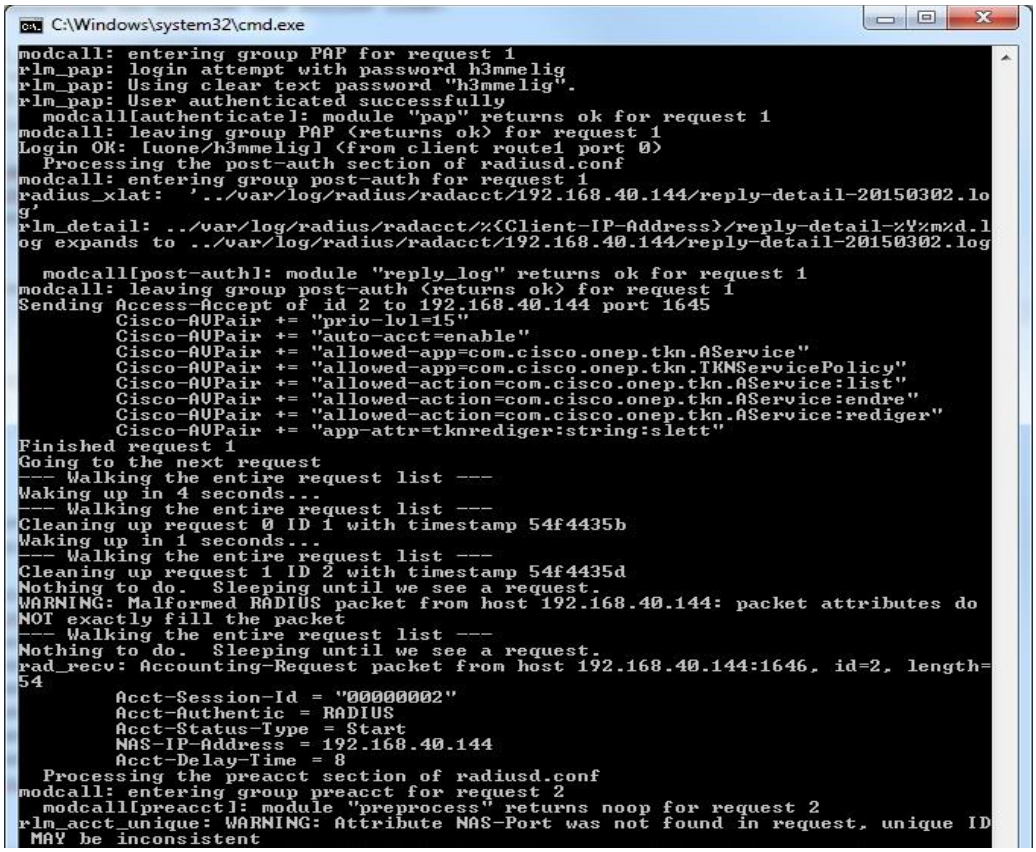
CPU, minneforbruk og buffer statistikk fra kjøring av AService på ruter 2:

Kommandoen `show onepk status <sesjon nummer>` viser status på sesjonen på ruteren:

```
Version: 1.3.0
Transport: tls; Status: running; Port: 15002; localcert: tknTP; client cert
validation enabled
Certificate Fingerprint SHA1: 609FFACC 811D5B9D 9FB55024 B1FB7226 49ACB6C3
Transport: tipc; Status: disabled
Session Max Limit: 10
CPU Interval: 0 seconds
CPU Falling Threshold: 0%
CPU Rising Threshold: 0%

(…)
```

RADIUS:

Brukergrensesnittet *debug mode* fra RADIUS når *uone* kjørte *AService*:



FF 2: Konsollvindu fra RADIUS

Autentisering:

```
Packet-Type = Access-Request
Mon Mar  2 12:02:51 2015

   User-Name = «uone»
   User-Password = «h3mmelig»
   NAS-IP-Address = 192.168.40.144

Client-IP-Address = 192.168.40.144
```

Autorisasjon:

```
Packet-Type = Access-Accept
Mon Mar  2 12:02:51 2015

   Cisco-AVPair += «priv-lvl=15»
   Cisco-AVPair += «auto-acct=enable»

    Cisco-AVPair += «allowed-app=com.cisco.onep.tkn.AService»

    Cisco-AVPair += «allowed-app=com.cisco.onep.tkn.TKNServicePolicy»

    Cisco-AVPair += «allowed-action=com.cisco.onep.tkn.AService:list»

    Cisco-AVPair += «allowed-action=com.cisco.onep.tkn.AService:endre»

    Cisco-AVPair += «allowed-action=com.cisco.onep.tkn.AService:rediger»

Cisco-AVPair += «app-attr=tknrediger:string:slett»
```

Regnskap (accounting) start:

```
Mon Mar  2 12:03:02 2015
   Acct-Session-Id = «00000002»
   Acct-Authentic = RADIUS
   Acct-Status-Type = Start
   NAS-IP-Address = 192.168.40.144
   Acct-Delay-Time = 8
   Client-IP-Address = 192.168.40.144
   Acct-Unique-Session-Id = «26339339d2e689a9»

Timestamp = 1425294182
```

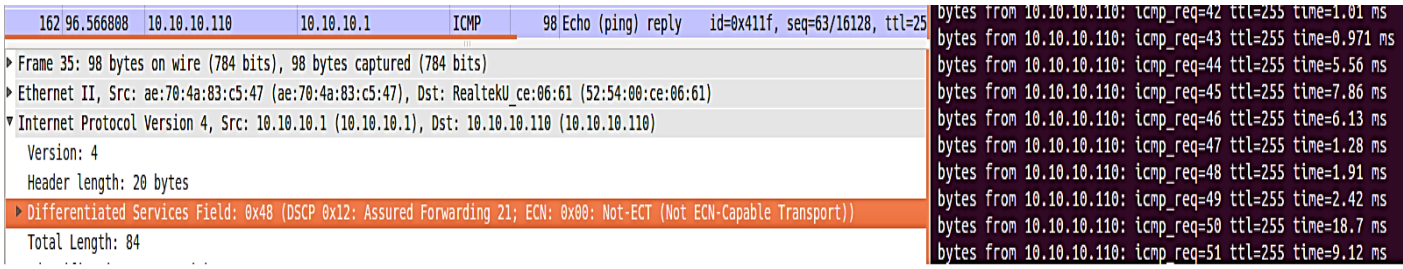
Parametere:

com.cisco.onep.element.SessionConfig		
public static final int	DEFAULT_EVENT_QUEUE_SIZE	2147483647
public static final int	DEFAULT_KEEPALIVE_IDLE_TIME	60
public static final int	DEFAULT_KEEPALIVE_INTERVAL	30
public static final int	DEFAULT_KEEPALIVE_RETRY_COUNT	2
public static final int	DEFAULT_PORT	15002
public static final int	DEFAULT_RECONNECT_TIMER	0
public static final int	DEFAULT_THREADPOOL_SIZE	10
public static final int	MIN_KEEPALIVE_IDLE_TIME	60

FF 3: Konstanter på nettverkselementet

Vedlegg G
Scenario: Militær prioritet for TKN
Noen utskrifter og logger fra tester

Et utdrag fra ping og analyse i Wireshark:



GF 1: Et utdrag fra Wireshark og ping

Kommandoen `show policy-map <interface>` viser hvilken *«QoS policy»* som kjøres på ruteren på et spesifikt grensesnitt.

QoS policy på IOP

Ingress grensesnitt ruter 1:

```
Router1# show policy-map interface gigabitEthernet 0/3

Service-policy input: __ONEP_4770_0
Class-map: __ONEP_4770_0 (match-any)
67 packets, 6566 bytes
5 minute offered rate 0000 bps, drop rate 0000 bps
Match: ip dscp cs4 (32)
67 packets, 6566 bytes
5 minute rate 0 bps
QoS Set
dscp af33
Packets marked 67
Class-map: __ONEP_4770_1 (match-any)
15 packets, 1470 bytes
5 minute offered rate 0000 bps, drop rate 0000 bps
Match: ip dscp af21 (72)
15 packets, 1470 bytes
5 minute rate 0 bps
QoS Set
dscp af23
Packets marked 15
(…)
```

Egress grensesnitt ruter 1:

```
Service-policy input: __ONEP_4770_1
Class-map: __ONEP_4770_8 (match-any)
70 packets, 6860 bytes
5 minute offered rate 0000 bps
Match: ip dscp af31 (26)
70 packets, 6860 bytes
5 minute rate 0 bps
Match: ip dscp af23 (22)
0 packets, 0 bytes
5 minute rate 0 bps

Class-map: class-default (match-any)
198 packets, 20280 bytes
5 minute offered rate 0000 bps, drop rate 0000 bps
Match: any
(..)
```

Minne og CPU forbruk av prosessen:

```
Router1#show onep statistics session all

Session ID: 4211

Application Name: TKNServicePolicy
API In: 18        API Out: 15
Bytes In: 3890          Bytes Out: 11463
Vty Count: 0

Memory Allocated: 1256704 bytes    Memory Freed: 1207296   Memory Held:
57408

CPU utilization for five seconds:  0.0 % one minute:  0.0 %      five
minutes:  0.0
```

Vedlegg H: Funksjonalitet fra InterfaceConN – en oversikt

I tabellen under er en beskrivelse av funksjonalitetene som er utviklet for Java klassen: InterfaceConN.

Funksjonalitet	**Beskrivelse**
Parser opsjoner	Sjekker opsjonene til programmet: Brukernavn, passord, node navn, evt. sertifikat (TLS). Hvis bruker ikke benytter noen opsjoner benyttes en standard fil, *tkn.egenskaper* som blant annet spesifiserer et grensesnitt. (For test formål)
Autentiseringsdialog	Viser en dialog boks for autentisering av bruker. Bruker må oppgi brukernavn og passord.
«Pinning» filområdet	Implementasjon av «TLS Certificate pinning[28]» Hvis TLS skal benyttes blir også sertifikater sjekket. Dette håndteres i klassen *TLSPH* som *InterfaceConN* kommuniserer med.
Grensesnitt funksjonalitet.	Setter opp TLS forbindelse. Henter status på grensesnitt. Henter Ethernett adresser. Henter brukernavn til bruker. Henter passord til bruker. Henter logisk adresse nettverksenhet. Fjerning av forbindelsen til nettverkselementet. Henter nettverkselementet. Henter egenskaper på grensesnitt. Lister ut alle grensesnitt på et nettverkselement. Lister ut status på et grensesnitt. Kjør opp/ned et grensesnitt. Viser autentiseringsdialog.

HT 1: Funksjonalitet i InterfaceConN

[28] Pinning er manuell autorisasjon og lagring av et pars offentlige nøkler for fremtidig kommunikasjon.