Web Services i nettverk med begrenset datarate

Dinko Hadzic, Trude Hafsøe, Frank Trethan Johnsen,
Ketil Lund og Kjell Rose

Forsvarets forskningsinstitutt (FFI)

21.12.2006

FFI-rapport 2006/03886

898

ISBN 978-82-464-1049-4

Emneord

Tjenesteorientert arkitektur (SOA)

Web Services

Nettverksbasert Forsvar (NbF)

Disadvantaged Grids

Komprimering

Meldingstjenesten i Forsvaret (MIF)

Militært meldingshåndteringssystem (MMHS)

Godkjent av

Ole-Erik Hedenstad	Prosjektleder
Vidar Andersen	Avdelingssjef

Sammendrag

Tjenesteorientert arkitektur (SOA) er identifisert som et av grunnprinsippene for Nettverksbasert Forsvar (NbF), og Web Services er i dag den vanligste teknologien for realisere SOA. Web Services muliggjør interoperabilitet mellom forskjellige systemer, men gir samtidig høyere overhead og dermed økte krav til datarate. Mens industrien kompenserer for dette med å utvikle nye kommunikasjonsteknologier med stadig høyere datarater, er situasjonen annerledes for radiosystemer i taktiske nett i Forsvaret, hvor militære krav til lang rekkevidde, sikkerhet, robusthet og beskyttelse mot jamming og avlytting ofte resulterer i begrenset datarate. Slike teknologier med lav datarate og ofte høy forsinkelse og pakketap kaller vi Disadvantaged Grids (DG).

I denne rapporten vurderer vi teknikker for å redusere overhead i XML og Web Services, og dermed gjøre det mulig å realisere SOA i nettverk med lave datarater. Våre målinger viser at det er mulig å oppnå mer enn 95 % reduksjon i datastørrelsen ved å bruke binær XML og komprimering. I tillegg har vi utført eksperimenter med Web Services over en rekke simulerte DG. Våre resultater viser at det er mulig å benytte Web Services også i nettverk med lave datarater, forutsatt at XML-data komprimeres og at man benytter en effektiv protokollprofil. Som databærer har vi benyttet et militært meldingshåndteringssystem (MMHS), med ulike protokollprofiler.

Til slutt diskuterer vi også nye mekanismer som kan være egnet for å distribuere informasjon til mottakere i et taktisk nett, og ser hvordan bruk av en proxy (mellomledd) kan bidra til å redusere overhead, gi høyere tilgjengelighet og senke forsinkelsen i forbindelse med kommunikasjon over nettverk.

English summary

Service-oriented architecture (SOA) is identified as one of the fundamental principles of Network Based Defence (NBD), and Web Services is a technology commonly used to realize SOA. Web Services promote interoperability between different systems, but at the same time increase the information overhead significantly, resulting in higher data rate demands. While the industry compensates for this by developing communication technologies with higher data rate, the situation is different for radio systems in military tactical networks where military requirements for long transmission range, security, robustness, anti-jamming and protection from wiretapping often result in limited data rate. We define such technologies with low data rate and often high delay and packet loss as Disadvantaged Grids (DG).

In this report, we consider methods to reduce the XML and Web Services overhead, in order to make it possible to realize SOA over DG. Our experiments show that it is possible to achieve more than 95 % reduction in size by using binary XML and data compression. In addition, we have performed experiments with Web Services over simulated DG. Our measurements show that it is possible to use Web Services even in networks with low data rates, as long as data compression and efficient communication protocols are employed. We have used a Military Message Handling System (MMHS) implementation, with different protocol profiles, as data carrier.

Finally we discuss new mechanisms that may be suitable for information distribution in tactical networks, and consider how proxies can contribute to reduce the overhead, increase the availability and lower the delay of network communication.

Innhold

1 Innledning

1.1 Bakgrunn

Tjenesteorientert arkitektur (SOA) er identifisert som et av grunnprinsippene i Nettverksbasert Forsvar (NbF), både av det norske Forsvaret [27] og NATO Network Enabled Capability Feasibility Study (NNEC FS) [30]. SOA vil benyttes for å realisere fleksible, utvidbare og interoperable systemer basert på løst koblede, autonome tjenester. Web Services er i dag den vanligste måten å realisere tjenesteorientert arkitektur på. Utfordringen er at Web Services har høy overhead, og gir dermed økte krav til datarate og prosesseringskraft. Industriens svar på problemstillingene er raskere og mer effektiv maskinvare, samt utvikling av nye kommunikasjonsteknologier som gir høyere datarater.

Situasjonen er annerledes i taktiske nett i Forsvaret. Radio systemer med fokus på lang rekkevidde, samt strenge militære krav til sikkerhet, robusthet og beskyttelse mot jamming og avlytting fører til at trådløse kommunikasjonsteknologier som brukes i Forsvaret ofte har svært begrenset datarate, sammenliknet med de sivile alternativene. Slike trådløse teknologier med lav datarate, høy forsinkelse og mye pakketap definerer vi som Disadvantaged Grids (DG).

Det er ønskelig å vurdere mulighetene for å bruke Web Services over DG. Fordelene er å kunne benytte standardiserte teknologier helt ut til taktisk nivå, samt gjenbruke eksisterende SOA-tjenester og applikasjoner. En vil også kunne oppnå en mer uniform informasjonsinfrastruktur (INI) som er enklere å realisere, administrere og bruke.

1.2 Mål

Målet med denne undersøkelsen er å vurdere mulighetene for bruk av Web Services over DG, både teoretisk og ved hjelp av praktiske eksperimenter og målinger. Vi presenterer og drøfter en rekke metoder som kan benyttes for å redusere overhead, som for eksempel komprimering, binær XML og omkoding til mer effektive formater.

Videre ønsker vi å identifisere nye mekanismer som kan være egnet for å distribuere informasjon til mottakere i et taktisk nett. Vi ønsker også å vurdere om bruk av en proxy kan bidra til å redusere overhead, gi høyere tilgjengelighet og senke forsinkelsen i forbindelse med kommunikasjon over nettverk.

1.3 Oppbygging av rapporten

Kapittel 2 presenterer et enkelt scenario som danner utgangspunkt for våre vurderinger og eksperimenter, samt gir en kort presentasjon av kommunikasjonsteknologier som vi har valgt å simulere i våre eksperimenter.

Kapittel 3 gir en introduksjon til Meldingstjenesten i Forsvaret (MIF), et system for sikker meldingsbasert kommunikasjon. MIF skal etableres som en gjennomgående infrastruktur, og skal støttes på flere nivåer i Forsvaret.

I kapittel 4 ser vi nærmere på teknikker for komprimering av XML-data, og vi presenterer og sammenlikner resultater fra praktiske målinger av komprimeringsrate og prosesseringstid.

Kapittel 5 diskuterer problemstillinger knyttet til bruk av Web Services over DG, og presenterer resultater fra målinger over simulerte taktiske kommunikasjonsteknologier.

Kapittel 6 fokuserer på metoder for en mer effektiv representasjon av informasjon, for å oppnå en reduksjon i mengden data som overføres, samtidig som håndteringen av dataene i endesystemene kan gjøres mer effektiv.

Kapittel 7 ser nærmere på selve utvekslingen av XML informasjon, det vil si hvilke prinsipper og mekanismer som kan være egnet for å distribuere informasjon til mottakere i et taktisk nett.

I kapittel 8 diskuterer vi hvordan en proxy kan benyttes til å optimalisere bruken av Web Services i nettverk med lave datarater, og foreslå noen teknikker som kan være aktuelle i et slikt scenario.

Til slutt gir vi i kapittel 9 en kort konklusjon av arbeidet som ble gjort. Her identifiserer vi også relevante problemstillinger for fremtidig arbeid.

2 Scenario

Vi har utarbeidet et enkelt scenario som danner utgangspunkt for våre vurderinger og eksperimenter. Formålet er å få en oversikt over hvilke interaksjonsformer som kan forekomme, og hva slags nettverkstyper som er i bruk. Dette scenariet dekker alle nivåer, selv om vårt fokus i all hovedsak vil ligge på taktisk og stridsteknisk nivå. Vi går også ut fra at all kommunikasjon er trådløs.

På strategisk og operasjonelt nivå opereres det normalt med relativt stor datarate, for eksempel vil kommunikasjonen mellom sentralt hovedkvarter og lokale kommandoplasser gjerne gå over radiolinje, som typisk tilbyr 6,5 Mbit/s. I tillegg kan man kommunisere med høy datarate over satellitt, mot installasjoner og fartøyer med stabiliserte antenner. Det finnes imidlertid også backup-løsninger, men da med lav datarate.

Mot kommandoplasser kan hovedkvarteret benytte satellitt, men siden vi antar at kommandoplassene kun disponerer utstyr med ustabiliserte antenner, vil dette gi svært begrenset datarate. Tilsvarende vil sentralt hovedkvarter kunne kommunisere med fartøyer over HF. Typiske systemer her er:

- STANAG 5066: Dette er hovedsakelig punkt-til–punkt samband, hvor radioen har et sett kanaler å velge mellom, og må prøve seg frem for å finne en som kan brukes (dette varierer med tid på døgnet, vær og så videre). Dermed blir linkens oppstartsforsinkelse svært stor, i verste fall opp mot 30 sekunder. Når kanalen først er satt opp kan man imidlertid sende data med relativt liten forsinkelse. Dataraten ligger typisk på 150 bit/s – 9,6 kbit/s.
- STANAG 4538 er en nyere standard, med samme bruksområde som 5066. Her er imidlertid kanalscanningen på sender- og mottakersiden synkroniserte, ved hjelp av GPS-klokker, noe som gjør det vesentlig raskere å finne en egnet kanal. Dataraten er tilsvarende som for STANAG 5066.

Mellom fartøyer vil man også kunne benytte Subnetwork Relay (SNR) som har en datarate på ca. 64 kbit/s. Her vil fartøyet med stabilisert satellittantenne fungere som en gateway mot omverdenen, som andre fartøy må gå via. Siden satellittforbindelsen vil ha vesentlig større datarate enn SNR-nettet, vil dette fartøyet i en del tilfeller måtte tilby proxy-funksjonalitet. Ved bruk av STANAG 5066/4538, vil ”gateway-fartøyet” også måtte ha en proxy-funksjon, men denne gangen for å tilpasse datamengden til den lave dataraten på HF-nettet.

På landsiden ser vi for oss en tilsvarende kommunikasjonsstruktur. Hovedkvarteret kommuniserer som nevnt med radiolinje til lokale kommandoplasser, evt. også med trange satellittkanaler. Kjøretøyene kan kommunisere direkte med hovedkvarteret over satellitt (lav datarate), og med kommandoplasser over High Band UHF (HB-UHF).

Både kommandoplasser og kjøretøyer kan kommunisere med lagførere over Lett Feltradio (LFR).

Denne tilbyr en maksimal datarate på 64 kbit/s, men konfigureres normalt til 2,4 kbit/s (i praksis oppnås det ikke mer enn ca. 1 kbit/s) for å oppnå tilstrekkelig rekkevidde. HB-UHF har på sin side en datarate på ca. 300 kbit/s, men samtidig dårligere rekkevidde. Dette gjør at det i mange tilfeller kan være aktuelt å bytte dynamisk mellom denne og MRR/LFR, slik at man utnytter dataraten i HB-UHF så lenge dette nettet er tilgjengelig, og så faller tilbake på HF-nettene når man mister kontakten.

Innad i et lag benyttes Personal Role Radio (PRR) for kommunikasjon, som tilbyr en datarate på 38,4 kbit/s. Det er viktig å merke seg at det kun er lagfører som kommuniserer med kjøretøy og/eller lokalt HQ, slik at øvrig personell må gå via lagfører, som dermed får en proxy-funksjon.

Som en oppsummering gir vi i Tabell 2.1 en oversikt over datarater for de ulike kommunikasjonsteknologiene som er i bruk i scenariet. Disse verdiene blir så brukt som grunnlag for oppsett av eksperimentene vi har utført.

	Datarate (Kbit/sek)
High band UHF (HB-UHF)	300,0
Subnetwork Relay (SNR)	64,0
Personal Role Radio (PRR)	38,4
STANAG 5066 **STANAG S4538**	9,6
Satellitt	2,4
Multirolleradio (MRR) **Lett Feltradio (LFR)**	1,0

Tabell 2.1: Datarater for ulike kommunikasjonsteknologier

3 Meldingstjenesten i Forsvaret (MIF)

I dette kapittelet gir vi en introduksjon til Meldingstjenesten i Forsvaret (MIF), et system for sikker meldingsbasert kommunikasjon som skal etableres som en gjennomgående infrastruktur i Forsvaret.

3.1 Introduksjon

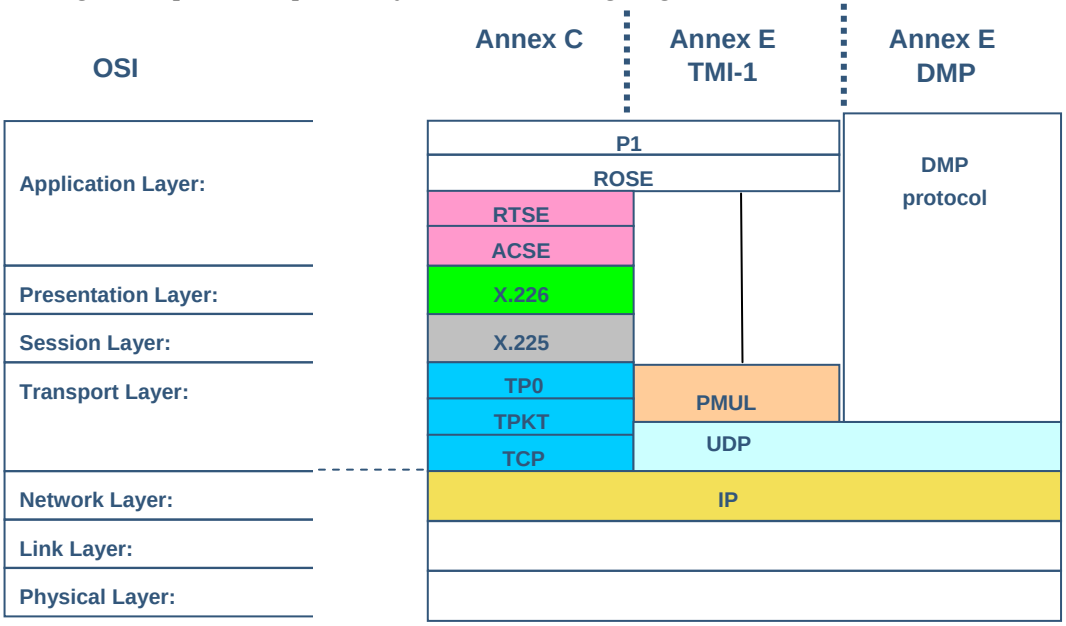
Forsvarsdepartementet har, som en del av INI-programområdet, satt i gang et prosjekt [29] som har som mål å etablere en gjennomgående militær meldingstjeneste i Forsvaret. Denne meldingstjenesten skal fungere over alle kommunikasjonssystemer Forsvaret har, fra strategisk til stridsteknisk nivå og mellom de ulike forsvarsgrenene. Ettersom deltagelse i internasjonale operasjoner utgjør en stadig større del av Forsvarets operasjoner er det viktig å basere en slik meldingstjeneste på NATOs standard for militære meldingshåndteringssystemer (MMHS), STANAG 4406. XOmail er en implementasjon av denne NATO-standarden, utviklet av Thales. I denne rapporten undersøker vi mulighetene for å benytte en slik meldingstjeneste som en bærer for Web Services ved å gjennomføre våre eksperimenter over XOmail.

3.2 Meldingsprofiler i MMHS

STANAG 4406 er en standard for et gjennomgående MMHS, og standarden definerer flere ulike protokollprofiler tilpasset ulike kommunikasjonssystemer. XOmail implementerer tre av disse protokollprofilene, en fra Annex C og to fra Annex E. Figur 3.1 viser hvilke protokoller som inngår i hver av disse protokollprofilene:

- **STANAG 4406, Annex C** definerer en protokollprofil som er utviklet med tanke på bruk i nettverk med høye dataoverføringsrater, og det er derfor lagt lite vekt på reduksjon av overhead og bytte av senderetning. Profilen benytter flere forbindelsesorienterte protokoller, og det må dermed opprettes en forbindelse på flere nivåer i protokollstakken før dataoverføringen kan påbegynnes. Dette medfører en rekke bytter av senderetning, noe som gjør profilen lite egnet til bruk over kommunikasjonssystemer hvor retningsbytter innebærer en betydelig forsinkelse av datatrafikken.
- **STANAG 4406, Annex E TMI-1** (Tactical Messaging Interface) definerer flere taktiske meldingsprofiler for ulike kommunikasjonssenarier. Den første av disse, i XOmail kalt TMI, er utviklet for å kunne benyttes over taktiske kommunikasjonssystemer, samtidig som den har full meldingsfunksjonalitet. Profilen er spesifisert av FFI, og benytter mer effektive protokoller som reduserer overhead og antall retningsbytter. En annen fordel med denne profilen er at den støtter bruk av multicast. Det at en melding kan adresseres til mange, men bare sendes én gang, kan redusere en applikasjons båndbreddebruk betraktelig. Denne profilen er pålitelig (effektiv retransmisjon), har komprimering av dataene som sendes, og den foretar fragmentering av meldinger der dette er nødvendig.
- **STANAG 4406, Annex E DMP** (Direct Messaging Profile) er en protokollprofil som nylig har blitt tatt inn i STANAG 4406, og den er beregnet brukt over taktiske kommunikasjonssystemer med svært begrenset kapasitet. Denne profilen er optimalisert

for minst mulig overhead og retningsbytter, og har derfor ikke full meldingsfunksjonalitet. Denne profilen gjenbruker ikke protokollene fra Annex C, men har skreddersydde formater som utveksles på de øverste lagene i protokollstakken for å oppnå størst mulig effektivitet, samtidig som den er kompatibel med de andre protokollprofilene ved bruk av en gateway. En ulempe med denne profilen er at den er beregnet for korte meldinger, og den har derfor ikke støtte for fragmentering. Dette gjør at største meldingstørrelse for meldinger som sendes over DMP er definert av datafeltet i nettlagsprotokollen, som for eksempel for IP er 64K. DMP kan konfigureres til å være enten pålitelig eller ikke. XOmails implementasjon av DMP er basert på en tidligere utgave av protokollspesifikasjonen for DMP, og følger derfor ikke standarden fullt ut.



Figur 3.1 Protokollprofiler i STANAG 4406

Tabell 3.1 viser omtrentlig overhead i bytes per pakke, samt antall retningsbytter for hver av de tre profilene. Høy overhead og mange retningsbytter gjør at Annex C-profilen er lite egnet til bruk over DG, og vi har derfor valgt ikke å ta med denne profilen i våre videre tester.

Protokollprofil	**Overhead (bytes)**	**Retningsbytter**
STANAG 4406 Annex C	2700	8
STANAG 4406 Annex E TMI-1	700	2
STANAG 4406 Annex E DMP	20	0 (1 ved pålitelig kommunikasjon)

Tabell 3.1 Overhead og retningsbytter for de ulike protokollprofilene

3.3 Fordeler med MMHS

I tillegg til at man kan benytte en allerede etablert infrastruktur, er det også i militære systemer hensiktsmessig å benytte en asynkron meldingsprotokoll som bærer for SOAP i stedet for den synkrone HTTP protokollen. Grunnen til dette er blant annet at request og respons meldinger kan

sendes uavhengig av ”acknowledgement”-mønsteret til HTTP og at informasjonen kan overføres over en serie av forskjellige kommunikasjonssystemer med ulike egenskaper (med hensyn til for eksempel forsinkelse, avbrudd og datarate) hvor det er behov for mellomlagring underveis. Andre fordeler ved å bruke meldingstjenesten:

- Store-and-forward: MMHS har støtte for store-and-forward, en teknikk som innebærer at meldinger som ikke kan leveres direkte til mottaker blir mellomlagret av andre noder inntil meldingen kan leveres.
- Prioritet og preemption: Meldinger som sendes kan tildeles en prioritet, og meldingene blir behandlet i henhold til denne. Dette innebærer at høyt prioriterte meldinger kan forsinke lavere prioriteter, men man oppnår samtidig lavere forsinkelse for viktig informasjon.
- Multicast: Ved bruk av en protokollprofil som støtter multicast kan antall meldinger som sendes gjennom nettverket ofte reduseres. Dette gjelder spesielt i radiobaserte DG med delte kanaler.
- Gjennomgående informasjonsutveksling: i INI prosjekt P8002 [29] vil Meldingstjenesten bli et etablert som en gjennomgående informasjonsutvekslingsmekanisme (helt ut på taktisk nivå), som det i en migrasjonsfase kan være hensiktsmessig å utnytte for å etablere en gjennomgående SOA.

4 Komprimering av XML-data

I dette kapittelet ser vi nærmere på teknikker for komprimering av XML-data. Videre presenterer og sammenlikner vi resultater fra praktiske målinger av komprimeringsrate og prosesseringstid for utvalgte teknikker.

4.1 Introduksjon

XML er det vanligste formatet som benyttes sammen med Web Services i dag. XML er et tekstbasert format for å beskrive data på en strukturert måte, og er godt egnet for utveksling av data mellom forskjellige systemer. XML inneholder både databeskrivelse (metadata) og data, representert i form av tekst som er leselig for mennesker. Utfordringen er at XML dokumenter har mye overhead, og normalt er en XML-representasjon av data mange ganger større enn en alternativ binær representasjon. I tillegg til datamengden, gir XML også overhead i prosesseringen; å prosessere og validere XML dokumenter er krevende for datamaskiner.

Komprimering av XML-data har mange potensielle fordeler, blant annet:

- Redusert lagringsplass. Dette gjelder både minne og persistent lagring (for eksempel databaser), og er spesielt viktig for små mobile enheter.
- Reduserte krav til datarate. Dette er en av de mest åpenbare fordelene. En vil kunne utnytte tilgjengelige kommunikasjonsressurser mer effektivt, samt anvende nettverk med lave datarater uten at dette går utover applikasjoner i særlig grad.
- Utvidet bruksområde for XML-baserte løsninger. XML applikasjoner vil også kunne benyttes i miljøer med lave datarater, for eksempel i taktiske systemer i Forsvaret.

Målet med denne undersøkelsen er å vurdere forskjellige teknikker for komprimering av XML-data. I dette kapittelet ser vi nærmere på tre forskjellige metoder som kan benyttes for komprimering av XML-data:

- Generell komprimering
- Binær XML
- Hybrider av binær XML og generell komprimering

Deretter presenterer vi resultatene fra de praktiske eksperimentene hvor vi har testet flere forskjellige komprimeringsteknikker, og målt komprimeringsrate og komprimeringstid for hver teknikk.

4.2 Komprimeringsteknikker

4.2.1 Generell komprimering

Generelle komprimeringsteknikker har som formål å redusere datamengden. Det komprimerte formatet egner seg derfor ikke for direkte manipulering, og man må først gjenskape originaldokumentet ved å dekomprimere det. Med hensyn til datakvalitet kan

komprimeringsteknikker deles inn i komprimering med og uten datatap. Eksempler på komprimering med datatap er JPEG og MPEG-4; etter komprimeringen kan man ikke rekonstruere opprinnelig datakvalitet. Disse komprimeringsteknikkene egner seg kun for data der man kan tolerere et visst kvalitetstap, som for eksempel ved komprimering av audio og video. De er ikke egnet for komprimering av XML og SOAP-meldinger. Eksempler på komprimering uten datatap er RLE, Huffmann og den mer effektive GZIP/ZLIB-algoritmen.

4.2.1.1 GZIP og ZLIB

GZIP (GNU ZIP) [7] er et åpent komprimeringsformat som er standardisert av IETF. GZIP er et velkjent format som er implementert på mange plattformer og støttes av mange programmeringsspråk. I tillegg benytter GZIP ingen patenterte løsninger.

ZLIB [8] er utviklet i etterkant av GZIP, men benytter nøyaktig den samme komprimeringsalgoritme internt. Vi utfører derfor tester kun med GZIP, men forventer svært like resultater som for ZLIB. Forskjellen er at ZLIB blant annet har en mer kompakt header og benytter en raskere sjekksum algoritme. Mens GZIP ble utviklet for å operere med filer, har ZLIB også et ”streaming” grensesnitt, som gjør det mer anvendbart for komprimering i nettverksapplikasjoner.

4.2.2 Binær XML

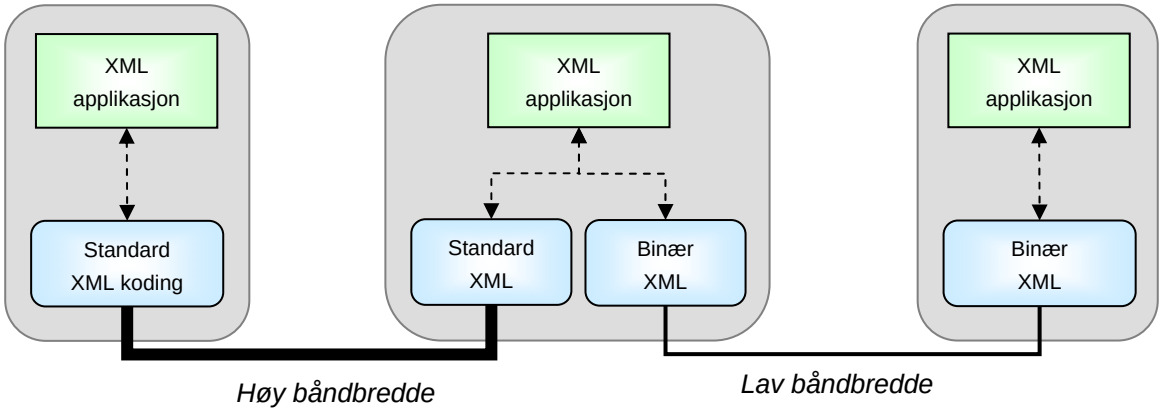
Mens XML representerer data i et tekstlig format som er leselig for mennesker, kan samme informasjonsinnhold kodes om til et binært format som er leselig for datamaskiner. Binær XML innebærer å omkode XML dokumentet fra et tekstformat til et binært format, en prosess som samtidig vil redusere datamengden betydelig. Det binære formatet vil fremdeles være et strukturert format, på samme måte som vanlig XML, og datamaskiner vil kunne direkte manipulere et binært XML format. Videre vil dette kunne føre til betydelig redusert transmisjonstid over kanaler med lave datarater, lavere minneforbruk og raskere prosessering.

I dag finnes det ingen offisiell standard for binær XML, og kommuniserende parter må bli enige om hvilken binær XML protokoll de skal benytte. W3C arbeidsgruppen Efficient XML Interchange (XMI) [1] har identifisert problemstillingen og jobber for å standardisere en binær XML teknikk. Det er derfor høyst relevant å følge med på dette arbeidet og resultatene som publiseres av denne W3C arbeidsgruppen.

Mens generell komprimering implementerer generiske teknikker som kan benyttes på alle typer dokumenter, opererer binær XML kun på XML dokumenter. Enkelte teknikker er skjemabaserte, mens andre er mer generiske og kan operere på alle gyldige XML dokumenter. Skjemabaserte teknikker tar utgangspunkt i et XML-skjema og gir typisk bedre komprimering enn ikke-skjemabaserte teknikker.

I motsetning til generelle komprimeringsmetoder er binær XML et strukturert format. Som eksemplifisert i Figur 4.1, kan en lage software som jobber direkte på det binære formatet, uten å måtte konvertere tilbake til tekstlig XML (XML-koding blir transparent for applikasjonene).

Ulempen er at det binære formatet er uleselig for mennesker, og dersom mennesker skal manipulere XML direkte må binær XML konverteres tilbake til standard (tekstlig) XML.



Figur 4.1 Binær XML er et strukturert format, og en kan utvikle applikasjoner som manipulerer det binære formatet direkte, på samme måte som standard (tekstlig) XML

Nedenfor gir vi en kort beskrivelse av de mest relevante implementasjonene av binær XML. Vi starter imidlertid med en presentasjon av et enkelt, binært referanseformat utviklet ved FFI, da dette gir en god beskrivelse av hvordan slike binære formater realiseres.

4.2.2.1 FFIs binære referanseformat

FFI har benyttet binære format ved sending av track-meldinger tidligere. Det egendefinerte referanseformatet stammer opprinnelig fra KKI-sjø-prosjektets demonstrator [25] hvor overføring av binære data over HF-radioer (kun HF-modem på laboratoriet) ble demonstrert. Formatet ble videre benyttet ved dataoverføring over Iridium under et eksperiment med distribuert bildeoppbygging under øvelsen Blue Game 2004. Disse formatene var ikke basert på XML.

Boolean	true															
Enumerations	friend															
Char	A															
Byte	123															
Short	123	45														
Int	123	45	67	89												
Long	123	45	67	89	01	23	45	67								
Float	12.	12	34	56												
Double	61.	23	45	67	89	01	23	45								
Complex types	1/0	123	45	67	89	123	45									
String	1/0		6	H	e	l	l	o	!							
List	1/0		2	1/0	123	45	67	89	123	45	1/0	123	45	67	89	123	45

Tabell 4.1 Et forsøk på å illustrere det binære referanseformatet. Hver rute i tabellen representerer 1 byte.

Referanseformatet genererer binære data automatisk ut fra tabeller som er avledet fra XML-skjemaet. Formatet er foreløpig ikke generelt og heller ikke komplett, da hensikten med det så langt kun har vært å understøtte prosjektets undersøkelser. Det egendefinerte referanseformatet er illustrert i Tabell 4.1.

De binære dataene overføres som et array av byte. De forskjellige datatypene overføres som følger:

- Boolske data overføres som 1 byte (false = 0, true = 1)
- Enumerations overføres som 1 byte (friend = 1, hostile = 2, o s v)
- Character overføres som 1 byte
- Byte overføres som 1 byte
- Short (korte heltall) overføres som 2 byte
- Integer (heltall) overføres som 4 byte
- Long (lange heltall) overføres som 8 byte
- Float (flyttall) overføres som 4 byte
- Double (flyttall med dobbel presisjon) overføres som 8 byte
- Komplekse eller sammensatte datatyper overføres med en byte som forteller om objektet er tomt eller ikke. Deretter overføres hvert av elementene i den sammensatte datatypen dersom objektet ikke er tomt
- Tekststrenger overføres med en byte som forteller om strengen er tom eller ikke. Deretter overføres lengden på strengen som et kort heltall (2 byte) og deretter overføres karakterene som strengen består av
- Lister av objekter

Et binært XML-format må kunne overføre alle data som er beskrevet av et lovlig XML-skjema. Det vil si at det binære formatet ikke skal være begrensende. Om ønskelig eller nødvendig kan formatet bli videreutviklet til å kunne representere et hvilket som helst XML-dokument som er beskrevet av et lovlig XML-skjema.

4.2.2.2 Fast Infoset (FI)

Fast Infoset (FI) [3][4] er en binær XML teknikk som er utviklet av Sun Microsystems og senere standardisert gjennom organisasjonene ITU-T og ISO. Fast Infoset er et binært format for representasjon av XML dokumenter, og er et alternativ til selve XML formatet. Målet med Fast Infoset er både å redusere den totale datamengden, og samtidig gjøre prosessering (serialisering og deserialisering) raskere og mer effektivt. Mens vanlige komprimeringsteknikker som GZIP kun optimaliserer dokumentstørrelsen, forsøker Fast Infoset å finne et optimalt kompromiss mellom komprimeringsrate og prosesseringshastighet. Mens Fast Infoset spesifikasjonen benytter ASN.1 syntaks for å gi en abstrakt beskrivelse av dataformatet, er det ingen krav om å benytte ASN.1 i implementasjoner av standarden.

4.2.2.3 Abstract Syntax Notation One (ASN.1)

Abstract Syntax Notation One (ASN.1) [5] er en standardisert og fleksibel notasjon for å beskrive data på en strukturert måte. ASN.1 er standardisert av ISO og ITU-T, og brukes blant annet for å

implementere telekommunikasjonsprotokoller på en effektiv og plattformuavhengig måte.

ASN.1 gir en syntaks for å beskrive data, men sier ikke noe om hvordan data skal kodes når de transmitteres over en kommunikasjonskanal. Det finnes derfor flere alternative kodingsregler for ASN.1:

- Basic Encoding Rules (BER) [10]
- Canonical Encoding Rules (CER) [10]
- Distinguished Encoding Rules (DER) [10]
- XML Encoding Rules (XER) [11]
- Packed Encoding Rules (PER) [12]
- Generic String Encoding Rules (GSER) [13]

Fire av kodingsreglene produserer et binært format (BER, CER, DER og PER), XER produserer et XML format, mens GSER gir et tekstlig format som er leselig for mennesker.

På mange måter kan ASN.1 sammenliknes med XML. Selve ASN.1 standarden gir en notasjon for å beskrive data som ville tilsvare et XML-skjema (XSD). Mens man instansierer et XSD-skjema i form av et XML dokument som overføres i klartekst, har ASN.1 har flere forskjellige kodingsformater, både tekstlige og binære. Sammenliknet med XML, er de binære ASN.1 formatene mer effektive, raskere å prosessere og gir betydelig redusert datastørrelse.

4.2.2.4 Efficient XML (EFX)

Efficient XML (EFX) [22] er en effektiv binær XML teknikk utviklet av AgileDelta. EFX kan komprimere alle XML dokumenter, både dokumenter som er beskrevet av et XML-skjema og frittstående dokumenter som ikke har et XML-skjema knyttet til seg. Komprimeringsresultater blir bedre dersom man benytter skjema-baserte XML dokumenter. AgileDelta er aktivt med i W3C gruppen som jobber for å få frem en binær XML standard [1], og vil oppdatere EFX periodisk for å implementere resultatene fra standardiseringsarbeidet. Når en binær XML teknikk er standardisert, vil EFX fullt ut støtte denne nye standarden.

4.2.3 Hybride komprimeringsmetoder

Hybride komprimeringsmetoder kombinerer flere teknikker for å oppnå best resultat. Ofte gir disse metodene gode komprimeringsresultater på bekostning av lengre komprimeringstid. For hver applikasjon må det derfor vurderes om den ekstra komprimeringsgevinsten er forsvarlig i forhold til den ekstra tiden som metoden bruker. Noen hybride teknikker er XML-spesifikke, og opererer kun på XML data, mens andre er mer generiske og aksepterer alle typer data som input. Til forskjell fra binær XML er ikke resultatet av komprimering med hybride teknikker et strukturert format som applikasjoner kan manipulere direkte.

4.2.3.1 Fast Infoset og GZIP

Ved å først konvertere XML dokumenter til Fast Infoset, og deretter komprimere resultatet med GZIP, kan en tenke seg å oppnå bedre komprimeringsrate enn med hver av teknikkene separat. Men samtidig som komprimeringsraten blir bedre, vil også den totale prosesseringstiden for

komprimering og dekomprimering øke.

4.2.3.2 XMill

XMill [9] er en teknikk som først restrukturerer XML dokumentet blant annet ved å gruppere liknende elementer, og deretter benytter ZLIB komprimering. XMill er utviklet av AT&T Research men er nå et open-source prosjekt som tilbyr en XMill implementasjon i C++.

4.3 Eksperimenter

I dette eksperimentet har vi har testet følgende komprimeringsteknikker:

- Fast Infoset
- GZIP
- Kombinasjon av Fast Infoset og GZIP
- XMILL
- Efficient XML (EFX) med bruk av skjema (XSD)
- Efficient XML (EFX) uten bruk av skjema (XSD)

Vi har benyttet standard (default) parametere til hver komprimeringsteknikk, og har ikke gjort forsøk på å optimalisere noen av disse. Som utgangspunkt for testene med komprimering har vi benyttet 25 XML filer som inneholder C2IEDM data, og varierer i størrelse tilnærmet lineært fra 72 bytes til 551 525 bytes. Resultatene fra dette eksperimentet vil variere for andre datasett, og bør derfor brukes kun som en generell indikator.

For hver teknikk måler vi følgende parametere, som vi senere sammenlikner og diskuterer:

- Komprimeringsrate
- Komprimeringstid
- Dekomprimeringstid

Hver komprimering/dekomprimering er gjentatt 100 ganger. For å fjerne svingningene i tid som kan forårsakes av for eksempel CPU trådskifte, bruker vi 90-persentilen som endelig resultat i stedet for gjennomsnittet.

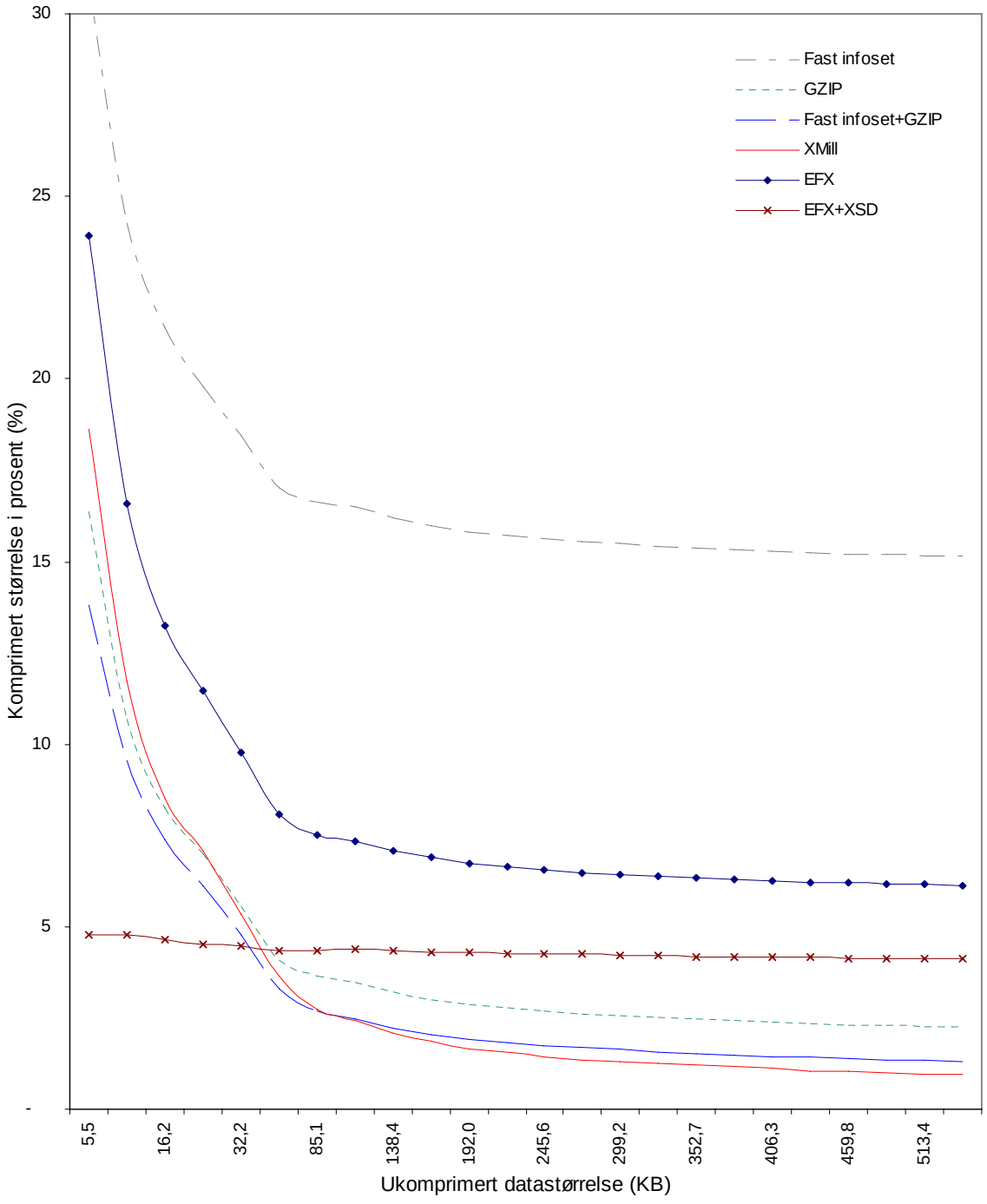
Eksperimentet og alle komprimeringsteknikkene er implementert i Java. Unntaket er XMill som er implementert i C++. På grunn av dette bør man ikke direkte sammenlikne XMills prosesseringstid mot de andre metodene, men kun bruke det som en generell indikator. Testene er gjennomført på Windows XP maskin med Pentium4 3,2 GHz CPU og 1,5 Gb minne.

4.4 Resultater

Alle teknikkene gir betydelig reduksjon i datastørrelsen, ofte mer enn 95 % reduksjon, men det er også betydelige forskjeller mellom teknikkene både når det gjelder komprimeringsrate og prosesseringstid. Figur 4.2 viser prosentvis komprimeringsrate for teknikkene som ble testet, mens Figur 4.3 og Figur 4.4 viser prosesseringstid for henholdsvis komprimering og dekomprimering.

4.4.1 Komprimeringsrate

Den minste filen vi har komprimert er på 72 bytes. Etter å ha komprimert denne med GZIP eller XMill, ble resultatet faktisk større enn 72 bytes. Dette skyldes en fast header som disse teknikkene har, som gjør at det normalt ikke er lønnsomt å komprimere små filer.



Figur 4.2 Prosentvis komprimeringsrate

Fast Infoset gir noe dårligere komprimering fordi teknikken er et kompromiss mellom komprimeringsrate og prosesseringshastighet. Målet med Fast Infoset er at applikasjoner skal kunne jobbe direkte med Fast Infoset dokumenter, og at dette skal være betydelig raskere enn å prosessere XML.

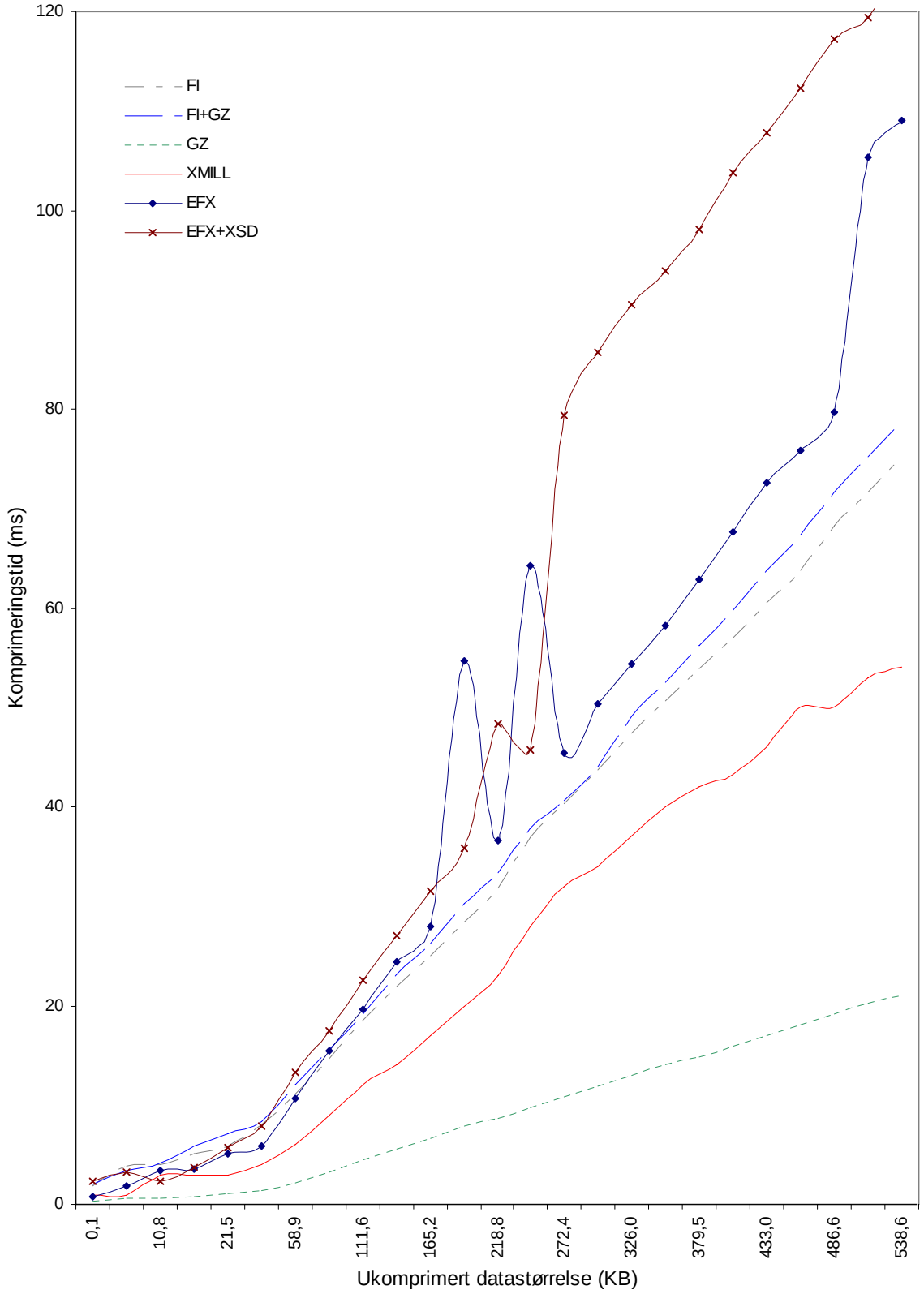
Videre ser vi at det er betydelig forskjell mellom de to Efficient XML (EFX) variantene. Den skjemabaserte varianten er mer effektiv, spesielt på filer mindre enn 30 KB, hvor teknikken er overlegen.

Det kan se ut som om forskjellen mellom de andre metodene er liten, men ser man nærmere er den relative forskjellen stor, spesielt for store filer. På en taktisk link med lav datarate kunne man spare flere sekunder ved å komprimere data med XMill eller en kombinasjon av Fast Infoset og GZIP, fremfor å benytte GZIP.

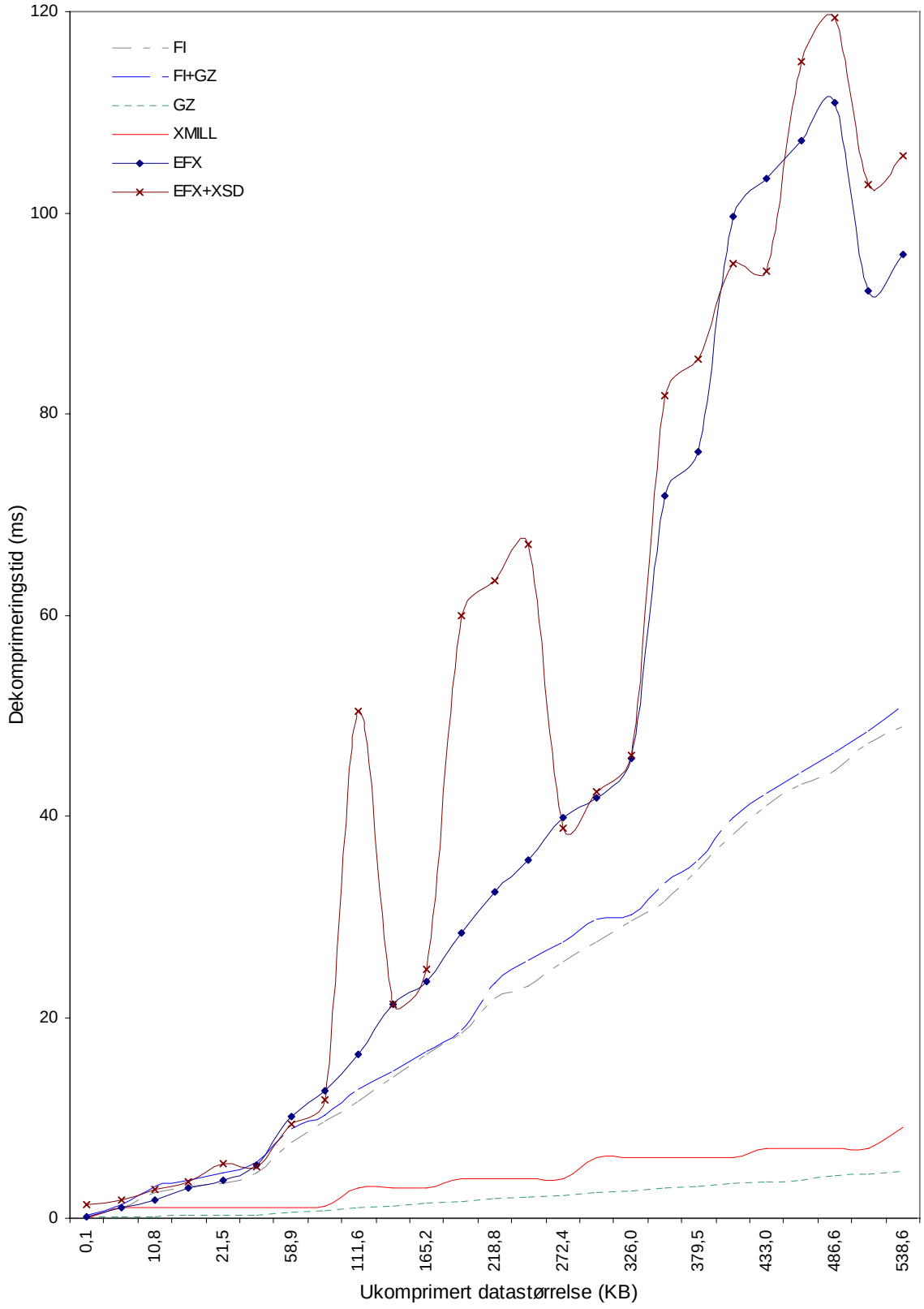
For filer større enn ca. 100 KB gir XMill best komprimering. I likhet med Fast Infoset og EFX, opererer XMill på XML-data. Resultatene indikerer at en kan oppnå bedre komprimering for store XML filer ved å velge en XML-spesifikk metode, fremfor en generell algoritme som GZIP.

4.4.2 Komprimeringstid

Ved å sammenlikne grafene i Figur 4.3 og Figur 4.4 ser vi at komprimering er mer tidskrevende enn dekomprimering, og dette gjelder for alle teknikkene vi har testet.



Figur 4.3 Komprimeringstid



Figur 4.4 Dekomprimeringstid

Til tross for at våre målinger er gjentatt 100 ganger og at vi har fjernet de største avvik ved å regne ut 90-persentilen for verdiene, ser vi at de to EFX metodene gir ujevne grafer i Figur 4.3 og Figur 4.4. EFX er en proprietær teknologi, og vi har ikke tilgang til komprimeringsalgoritmen

eller kildekoden. Vi antar derfor at ujevnhetene kan skyldes interne buffere som blir overfylt og må reinitialiseres.

For systemer med mye ledige ressurser (CPU og minne) og lite trafikk vil prosesseringstiden for komprimering og dekomprimering være akseptabel for alle teknikkene. Tiden vi sparer ved å overføre en komprimert fil over taktiske linker er mye høyere sammenliknet med tiden vi investerer i komprimering og dekomprimering. Men samtidig vil dette være en viktig faktor i systemer som er belastet med mye trafikk. Et annet moment er at radioer eller andre enheter i felt som går på batteri vil bruke mer strøm hvis prosesseringstiden for komprimering og dekomprimering er lang, noe som vil føre til redusert batterilevetid.

GZIP er den raskeste teknologien både for komprimering og dekomprimering. Vi ser også at den relative forskjellen mellom teknikkene er stor, og dette må tas i betraktning når man skal implementere systemer som skal håndtere store trafikkmengder. La oss se på et eksempel basert på våre målinger: vi skal sende ut meldinger på 300 KB som vi først skal komprimere. GZIP vil klare å komprimere ca. 80 slike meldinger i sekundet, Fast Infoset vil klare ca. 20 i sekundet mens den skjemabaserte EFX varianten kun klarer ca. 10 meldinger i sekundet.

4.5 Oppsummering

Undersøkelsen viser at det er mye å hente ved å komprimere XML-data, som har mye overhead og dermed er ”komprimeringsvennlig”. Flere algoritmer gav en reduksjon i datastørrelsen på mer enn 95 %. Formålet med denne undersøkelsen er ikke å velge én metode som er best, ettersom valget vil være avhengig av kravene som stilles i hvert enkelt scenario. GZIP er en god kandidat i mange sammenhenger: metoden er rask, tilbyr god komprimering og finnes fritt tilgjengelig på mange plattformer. Tilsvarende konklusjon gjelder også ZLIB, som benytter samme algoritme for komprimering. GZIP og ZLIB opererer på alle datatyper og er ikke begrenset til XML, i motsetning til de andre teknikkene vi har testet.

Våre målinger indikerer likevel at en kan oppnå bedre komprimering ved å velge en XML-spesifikk metode som Efficient XML eller XMill, fremfor en generell algoritme som GZIP. Vi legger også merke til at XML-spesifikke metoder har lenger prosesseringstid enn GZIP.

5 Web Services over Disadvantaged Grids

I dette kapittelet diskuterer vi problemstillinger knyttet til bruk av Web Services over DG, og presenterer resultater fra målinger over simulerte taktiske kommunikasjonsteknologier.

5.1 Introduksjon

Web Services og XML er i dag den vanligste måten å realisere tjenesteorientert arkitektur (SOA) på. Mens disse teknologiene løser mange problemer knyttet til interoperabilitet og metadata, fokuserer de ikke på effektivitet og datamengden som faktisk utveksles. XML dokumenter har mye overhead og resulterer ofte i datastørrelser som er mange ganger større enn et alternativt binært format. Ved å kompensere med kraftigere maskinvare og høyere datarater i Internett, er ulempene til XML overhead ofte akseptable når en tar i betraktning fordelene som Web Services og XML gir.

I Forsvaret er situasjonen annerledes, der trådløse taktiske kommunikasjonsteknologier ofte har svært begrenset datarate og høy forsinkelse og mye pakketap, sammenliknet med sivile alternativer. Tradisjonelt har Forsvaret valgt rekkevidde, kommunikasjonssikkerhet, robusthet, og beskyttelse mot jamming og avlytting fremfor å maksimere tilgjengelig datarate. Lave datarater fører til at kommunikasjon tar lang tid, og tid er ofte faktoren som bestemmer om Web Services kan brukes i sammenknytning med DG, eller ikke. Selv med en pålitelig link, er det i mange applikasjoner ikke akseptabelt med lange responstider, og dette kan gjelde både på bruker- og protokollnivå. I denne undersøkelsen anser vi derfor responstiden for en kommunikasjonssekvens (round-trip time, RTT) som en viktig parameter.

Formålet med denne undersøkelsen er å identifisere problemstillinger og mulige løsninger knyttet til bruk av Web Services over DG. Vi har simulert diverse DG linker med varierende datarate og forsinkelse, for å kunne identifisere grenseverdier for hva som er realistisk å få til med kommunikasjonsutstyr som eksisterer i Forsvaret i dag.

5.2 Mapping til XOmail

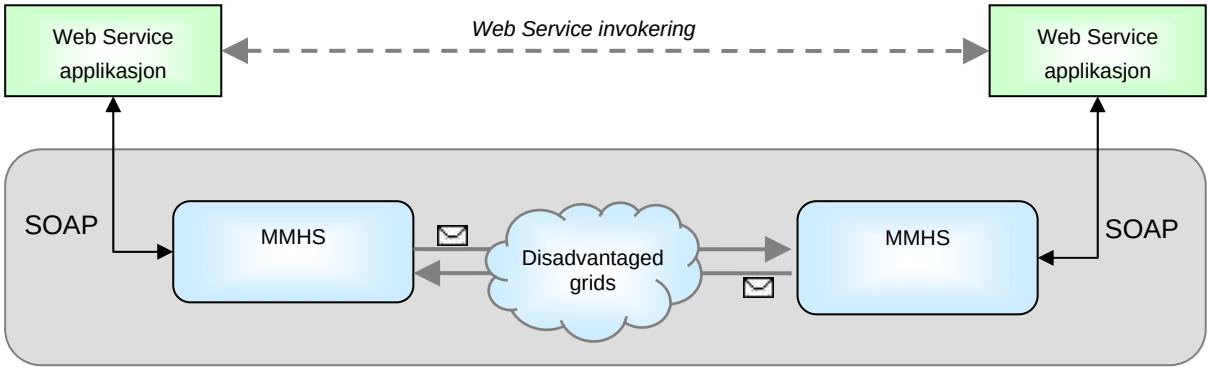
I de fleste Web Services applikasjoner benyttes HTTP for å overføre SOAP meldinger. HTTP faller naturlig inn i request/response modellen som ofte benyttes i Web Services, hvor SOAP request pakkes inn i HTTP request meldinger, mens SOAP response pakkes inn i HTTP response meldinger.

En fordel med SOAP er at den ikke er begrenset til en spesiell transportprotokoll, spesifikasjonen [14] åpner for bruk av andre transportprotokoller som SMTP (Simple Mail Transfer Protocol) eller FTP (File Transfer Protocol). Tanken er at hvis meldingsoverføringsprotokollen SMTP kan benyttes, så kan også andre meldingssystemer benyttes, som for eksempel MMHS.

XOmail er en implementasjon av STANAG 4406, og tilbyr sikker, meldingsbasert

kommunikasjon, som er uavhengig av underliggende transmisjonsteknologi. Ved å overføre SOAP over XOmail kan vi utnytte store-and-forward meldingsformidling for å realisere asynkron transport for SOAP meldinger. Fordelen er at vi i tillegg til den klassiske HTTP request/response modellen kan realisere asynkrone tjenester som publish/subscribe og multicast av SOAP. Vi kan også enklere benytte Web Services over DG, blant annet fordi vi unngår HTTP timeout. I tillegg sørger XOmail for informasjonssikkerhet på transportlaget samt adressering, ruting og prioritering av pakker.

I våre eksperimenter benytter vi XOmail, en MMHS implementasjon, for å transportere SOAP mellom Web Service applikasjoner, som vist i Figur 5.1. SOAP overføres som et MIME (Multipurpose Internet Mail Extensions) vedlegg til selve MMHS-meldingen.



Figur 5.1 Meldingstjenesten fungerer som et transportlag for Web Services. Meldingstjenesten sørger for adressering og sikker transport av SOAP meldinger.

5.2.1 Konfigurasjon av protokollprofilene

Før bruk må XOmails protokollprofiler konfigureres til å ta hensyn til hva slags type kommunikasjonssystem som skal benyttes. Noen av disse konfigurasjonsparameterne kan påvirke resultatene av en effektivitetsmåling, og profilkonfigureringen må derfor tas med i betraktningen når protokollprofilene skal vurderes:

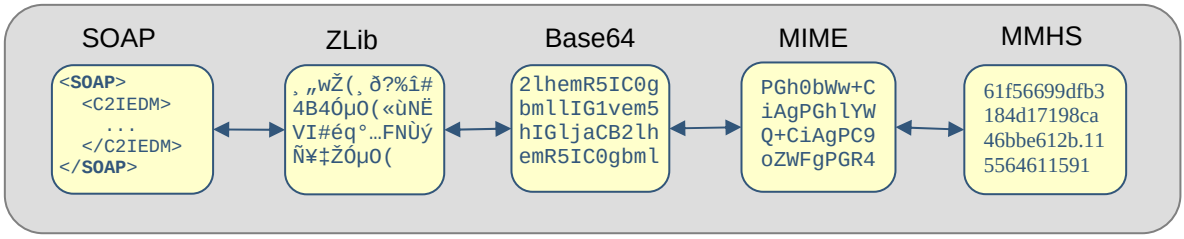
- En viktig konfigurasjonsparameter er pakkestørrelsen som benyttes. Denne innstillingen regulerer forholdet mellom hvor mye som sendes per pakke i forhold til hvor mange pakker en melding deles opp i. Dersom denne verdien er sett for lavt, vil en stor del av tilgjengelig datarate gå med til overhead. En for høy verdi for pakkestørrelsen vil medføre at den lange tiden det vil ta å sende pakken over nettverket, stopper viktigere trafikk fra å bli overført. I våre tester har vi brukt en pakkestørrelse på 32K for TMI og standardinnstilling for DMP.
- TMI er utviklet med tanke på å kunne benyttes over radiosystemer. Disse systemene har liten evne til å mellomlagre store mengder data, og det er derfor viktig ikke å sende for mye informasjon over et slikt system på én gang. Noen radiosystemer unngår dette problemet ved å benytte seg av ICMP Source Quench, en teknikk som lar mottakeren gi senderen beskjed om å redusere senderaten. Ikke alle radiosystemer støtter Source Quench, og for å unngå problemet med å sende for mye informasjon også over slike systemer, har TMI en standardinnstilling som sier at protokollen skal vente tre sekunder

mellom hvert pakkefragment som sendes. En tre sekunders forsinkelse er lite hensiktsmessig når man skal teste kommunikasjonssystemets begrensinger over et nettverk som ikke har dette problemet, og vi har derfor satt ned denne forsinkelsen til den lavest mulige verdien, nemlig ett millisekund.

- DMP kan konfigureres til enten å være pålitelig eller upålitelig. Når pålitelig overføring brukes vil kvitteringsmeldinger bli benyttet for å sikre at all informasjon har kommet frem til mottakeren. Disse kvitteringsmeldingene vil konkurrere om den tilgjengelige dataraten med annen datatrafikk, og vil i tillegg medføre flere bytter av senderetning. I den standardiserte utgaven av DMP håndteres kvitteringsmeldinger ved hjelp av en vindusprotokoll, og man kan derved ha flere utestående meldinger før kvitteringmeldingen sendes. På denne måten fylles kanalen opp og unødige retningsbytter unngås. XOmail-implementasjonen av DMP sender derimot en kvitteringsmelding per mottatt pakke, noe som genererer langt flere kvitteringsmeldinger enn hva standarden tilsier. Dette medfører at mengden brukerdata som kan overføres reduseres ved pålitelig kommunikasjon reduseres mer av XOmail-implementasjonen enn hva standarden skulle tilsi, og våre eksperimenter er derfor utført med upålitelig overføring.

5.2.2 Dataflyt

Meldingstjenesten bør i størst mulig grad være transparent for Web Service applikasjonene, og det bør fremstå som et transport- eller mellomvarelag som sørger for at pakker kommer frem. Som illustrert i Figur 5.2, vil dataene gjennomgå flere transformasjoner mellom forskjellige dataformater på sin vei mellom to Web Service applikasjoner. Web Service applikasjoner leverer SOAP pakker til meldingstjenesten, som vil komprimere med ZLIB algoritmen og konvertere det binære resultatet til Base64. Base64 benyttes ofte for å overføre binære data representert i form av leselige bokstaver. Bruk av Base64 resulterer i økt datastørrelse, fordi det binære formatet nå representeres i et tekstformat. Base64 data legges deretter inn i en MIME melding, som blir et vedlegg til MMHS pakken.



Figur 5.2 Dataflyt i eksperimentet: Java objektene serialiseres til SOAP meldinger, som komprimeres av XOmail med ZLIB algoritmen. Deretter kodes dataene om til Base64 format, før de transporteres som et MIME vedlegg i en MMHS pakke.

5.3 Effektiv transmisjon av SOAP meldinger

I dette avsnittet skisserer vi forskjellige teknikker som kan benyttes for å gjøre transmisjon av SOAP mer effektiv, samt redusere overhead i SOAP meldinger.

En åpenbar mulighet er å komprimere hele SOAP meldingen og transportere det komprimerte

resultatet. Denne teknikken er effektiv, men forutsetter at sender og mottaker er enige om at innholdet skal komprimeres og med hvilken teknikk, uten at man kan beskrive dette i grensesnittet i en WSDL fil. Mellomvare eller transportlag som støtter komprimering (slik som XOmail) løser problemet ved å utføre komprimering på transportlaget i stedet for applikasjonslaget, som resulterer i at komprimering blir usynlig for Web Service applikasjonen.

En annen mulighet er å overføre innholdet SOAP body som en komprimert SOAP vedlegg, mens SOAP header overføres i klartekst. Fordelen er at en tjenestens grensesnitt (definert i WSDL) kan beskrive de binære data, samtidig som SOAP header informasjon fortsatt er tilgjengelig i klartekst. For å få dette til kan en benytte teknikken Message Transmission Optimization Mechanism (MTOM) [20], som er utviklet av W3C. MTOM gjør det mulig å overføre binære objekter som MIME vedlegg til SOAP meldinger.

Videre er det også mulig å benytte Fast Web Services [6]. En må ikke forveksle denne standarden med Fast Infoset, som er en binær XML teknikk. Fast Web Services standarden gjør det mulig å beskrive SOAP meldinger med ASN.1 notasjon, og deretter kode med en av ASN.1 kodingsformatene som for eksempel PER [12]. Meldingene vil fremdeles ha samme innhold og semantikk som W3C SOAP meldinger, selv om de er representert i et binært format. Applikasjoner basert på Fast Web Services vil kreve mindre datarate og kunne være raskere å prosessere enn ”klassiske” Web Services basert på XML.

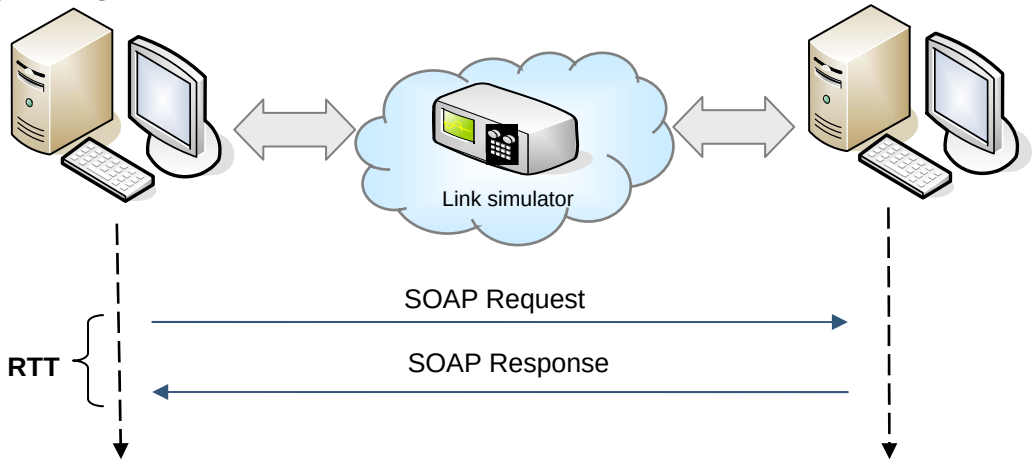
5.4 Eksperimenter

Vi har benyttet en link simulator [2] for å simulere flere DG teknologier, som oppsummert i Tabell 5.1. Vi simulerer kun typiske gjennomsnittlige datarater til ulike taktiske linker. Andre faktorer som vil spille inn ved bruk i felt er nodenes mobilitet og de variasjoner i signalstyrke, tilgjengelig datarate og feilrate dette vil medføre. Undersøkelsene er ment som en pekepinn på hvorvidt det er mulig å innføre Web Services på taktisk nivå, og skal ikke være en uttømmende undersøkelse utført med tanke på deployering. Vi simulerer kun én link, noe som innebærer at vi undersøker overføringstiden over ett transmisjonsmedium av gangen. Med de ressursene vi har tilgjengelig kan vi ikke utføre eksperimenter med flere linker (og dermed flere hopp). Slike undersøkelser vil være en nødvendighet som en del av framtidig prototyping.

	Datarate (Kbit/s)	**Forsinkelse (ms)**
High band UHF (HB-UHF)	300,0	0
Subnetwork Relay (SNR)	64,0	0
Personal Role Radio (PRR)	38,4	0
STANAG 5066 **STANAG 4538**	9,6	0
Satellitt	2,4	1500
Multirolleradio (MRR) **Lett Feltradio (LFR)**	1,0	0

Tabell 5.1 Datarate og forsinkelse for kommunikasjonsteknologier som ble simulert

Videre har vi implementert en Web Service som produserer SOAP meldinger med varierende størrelser. Vi måler da round-trip time (RTT), som er tiden fra vi sender en SOAP request til vi har mottatt en SOAP response. Oppsettet er illustrert i Figur 5.3. Målingene er gjort med TMI og DMP profilene som begge er basert på UDP, og begge profilene implementerer ZLIB komprimering.



Figur 5.3 Lab-oppsett for Web Service RTT-målinger

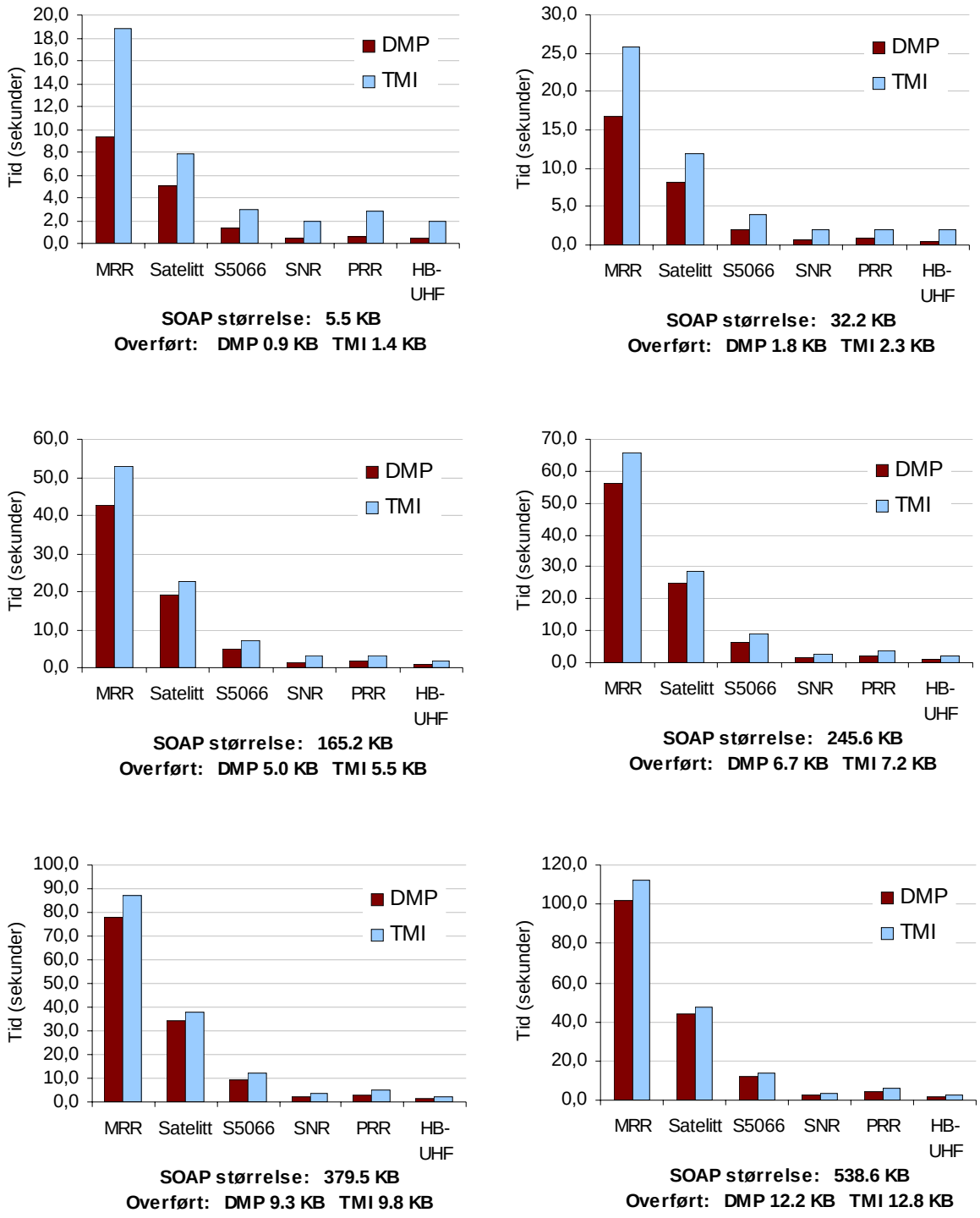
XOmail er basert på store-and-forward teknikken, og implementasjonen tilbyr en rekke konfigurasjonsmuligheter av denne mekanismen, for eksempel bufring av pakker, som igjen kan føre til forsinkelse internt i XOmail. Fordi vi er interessert i å måle transmisjonstiden har vi konfigurert XOmail med tanke på minimal forsinkelse internt i systemet.

5.5 Resultater

Figur 5.4 oppsummerer resultater fra målinger av Web Services over DG. Hver graf i figuren viser RTT (round-trip time) for en Web Service eksekvering, og vi kan sammenlikne de seks simulerte kommunikasjonsteknologier, og de to MMHS profiler (DMP og TMI).

Figuren viser grafer for seks forskjellige SOAP størrelser. For hver graf ser vi SOAP størrelsen som Web Service applikasjonen sender, samt datamengden som faktisk blir overført av DMP og TMI. Den overførte datamengden blir betydelig mindre enn SOAP størrelsen, og dette skyldes at begge profilene implementerer ZLIB komprimering. Vi ser klart at komprimering spiller en viktig rolle for å redusere datamengden som skal transmitteres over linker med lav datarate. For eksempel, en SOAP melding på 538,6 KB blir redusert til 12,2 KB ved bruk av DMP, som er en reduksjon på hele 97,7 %.

Målingene bekrefter også at DMP er svært båndbreddeeffektiv, ettersom DMP er raskere enn TMI i alle testene som ble utført.



Figur 5.4 RTT målinger for Web Services over Disadvantaged Grids

5.6 Oppsummering

Undersøkelsen viser at det er teknisk mulig å benytte MMHS som bærer av SOAP meldinger, og på denne måten realisere bruk av Web Services i taktiske nett. Den andre utfordringen er begrenset datarate. Vi valgte derfor å simulere seks forskjellige kommunikasjonsteknologier som

brukes i Forsvaret, og kjørte deretter Web Services over disse simulerte linker og målte eksekveringstiden i hvert tilfelle.

Formålet med denne undersøkelsen er ikke å identifisere teknologier som kan og ikke kan brukes sammen med Web Services, da dette vil være forskjellig for hvert system og hver applikasjon. MRR og satellitt vil trolig være lite egnet til bruk i interaktive applikasjoner hvor en bruker er involvert, men vil likevel kunne benyttes i bakgrunnsprosesser. De andre simulerte teknologiene vil være mer egnet også for interaktive applikasjoner. Uansett tilgjengelig datarate og SOAP størrelse bekrefter våre målinger betydningen av datakomprimering.

6 Effektiv representasjon av informasjon

I kapittel 4 diskuterte vi hvordan datamengden som overføres kan reduseres gjennom komprimering av dataene. Det er imidlertid også viktig å se på selve informasjonen som overføres, og forsøke å optimalisere representasjonen av denne. På den måten oppnår man en ytterligere reduksjon i mengden data som overføres, samtidig som håndteringen av dataene i endesystemene kan gjøres mer effektiv. Det siste gjelder særlig krav til minneforbruk og prosesseringskraft i forbindelse med serialisering og deserialisering av objekter.

6.1 Multilateral Interoperability Programme (MIP)

Multilateral Interoperability Programme (MIP) har lenge arbeidet med det som ble til ”Command and Control Information Exchange Data Model” (C2IEDM) og som senere har blitt videreutviklet til ”Joint Command, Control and Consultation Information Exchange Data Model” (JC3IEDM). Disse modellene er i utgangspunktet beregnet for utveksling av data mellom relasjonsdatabaser, og målet med arbeidet er å oppnå internasjonal interoperabilitet for kommando- og kontrollinformasjonssystemer (K2IS) på alle nivåer.

JC3IEDM er svært omfattende, og inneholder nå over 240 entiteter, som dekker blant annet type, posisjon, status, assosiasjoner og tilhørighet for objekter (materiell og organisasjoner), og det er rapporteringsdata tilknyttet all annen informasjon, for å muliggjøre sporing. Videre er det også et stort antall relasjoner (fremmednøkler) mellom ulike entiteter, i mange tilfeller også flere relasjoner mellom de samme entitetene. Til sammen gjør dette at strukturen i modellen blir svært kompleks.

Samtidig har det også blitt designet en objekt-orientert versjon av modellen (heretter kalt OO-XML), for å tilpasse modellen til en SOA-verden. Blant annet fordi modellen i utgangspunktet var svært kompleks, har det imidlertid vist seg at denne OO-XML versjonen av datamodellen representerer en lite effektiv måte å utveksle informasjon på. I forbindelse med eksperimenter gjort under CWID[1] 2006, hvor vi benyttet en redusert utgave av C2IEDM, fant vi for eksempel at bare for et skip å rapportere om egen posisjon, type og status, krevdes det mer enn 250 linjer med XML-kode. Det burde være klart at dette er uheldig i DG-sammenheng, selv om man benytter komprimering.

Til tross for den dårlig effektiviteten til OO-XML versjonen av JC3IEDM, burde den kunne fungere som et utgangspunkt for design av en mer effektiv datautvekslingsmodell. Vi har allerede gjort en innledende studie rundt dette [23] og har der redusert størrelsen på dokumentene med over 60 %. Dette har vi i hovedsak oppnådd ved å fjerne listestrukturer, og ved å redusere redundans, først og fremst i forbindelse med rapporter.

[1] Coalition Warrior Interoperability Demonstration

6.2 NATO Friendly Force Tracking (NFFI)

Som nevnt over er den nåværende MIP Web Services/Objekt-orienterte XML datautvekslingsmodellen lite effektiv, og Friendly Force Tracking (følging av vennlige styrker) er nettopp et resultat av at modellen ikke sees på som velegnet for anvendelse over DG. NATO har her laget en alternativ datautvekslingsmodell for følging av egne styrker i Afghanistan, NATO Friendly Force Information (NFFI) Afghanistan Force Tracking System (AFTS). Den nåværende versjonen av standarden er NFFI 1.3 som også er publisert som Draft STANAG 5527 [25]. NFFI består av meldingsdefinisjonen og meldingsprotokoll.

Meldingsprotokollen er spesifisert som to profiler som begge ble testet på CWID 2006:

- IP1 som er basert på TCP
- IP2 som er basert på UDP

Det arbeides også med nye profiler basert på Web Services standarder.

Meldingsformatet er definert i form av et XML-skjema. Det inneholder felter for posisjonsdata, identifikasjonsdata, status etc. Posisjonsdataene inneholder posisjon og hastighet, og fordi dette er det eneste obligatoriske feltet er også identifikasjonsdata (tilsvarende en trackId) lagt under dette feltet. Feltet for identifikasjonsdata inneholder navnet til objektet, og en 15 karakterers tekststreng fra APP-6A/Mil STD 2525B som sammen med posisjonsinformasjonen innholder all informasjonen som er nødvendig for å tegne et symbol på et kart. Statusfeltet inneholder den operative status. De andre feltene er valgfrie og inneholder informasjon som kontaktinformasjon, telefonnummer etc. Meldingsformatet er foreløpig begrenset til følging av vennlige styrker, men kan utvides til å omfatte alle aktører i operasjonsområdet.

Det er et krav til NFFI at formatet skal kunne oversettes til MIP C2IEDM, og i spesifikasjonen av standarden er mappingen av feltene i NFFI til felter i C2IEDM identifisert.

Denne versjonen av NFFI spesifiserer ikke komprimering, men våre måleresultater har vist at dette kunne redusert datamengden til ca 5 % av størrelsen av XML-dokumentet, når flere relativt like track rapporteres i den samme meldingen.

6.3 En sammenligning av MIP og NFFI

MIP skal dekke hele spekteret av kommando og kontroll, og modellen må nødvendigvis være omfattende. Det tar lang tid å sette seg inn modellen og implementere den. NFFI derimot er enkel, og det er fort gjort å sette seg inn i modellen.

Dersom MIPs JC3IEDM/C2IEDM hadde hatt et optimalt XML-format hvor all unødvendig informasjon var skrellet bort eller var valgfri, ville det antakelig ikke ha vært behov for et eget NFFI meldingsformat. Men dagens versjon av MIPs Web Services/Object-Oriented XML-format som ble benyttet ved målingene i kapittel 5.4 anses å være mindre egnet til bruk over DG. FFI har derfor foreslått forbedringer til det eksisterende formatet [23], noe som vil gjøre MIP formatene

adskillig mer anvendelige i en SOA-sammenheng. Før man får på plass et effektivt XML format for JC3IEDM, er det grunn til å regne med at alternative modeller vil bli valgt ved realisering av SOA.

7 Informasjonsutveksling

I tillegg til en effektiv representasjon av informasjonen er det viktig å se på hvordan selve utvekslingen av data gjøres, slik at man kan utnytte egenskapene i det underliggende nettverket best mulig. I dette kapittelet ser vi derfor nærmere på hvordan informasjonen representeres og utveksles, det vil si hvilke prinsipper og mekanismer som kan være egnet for å distribuere informasjon til mottakere i et taktisk nett. Ettersom arbeidet vårt hittil har vært konsentrert rundt datamodellene som er utviklet innenfor MIP, C2IEDM og JC3IEDM, vil disse modellene bli brukt som basis i dette kapittelet. Vurderingene som gjøres er likevel ikke begrenset til disse modellene. Merk også at det er en underliggende antakelse at det er XML-baserte data som utveksles.

7.1 Informasjonsutveksling i MIP-sammenheng

C2IEDM og JC3IEDM støtter tre ulike mekanismer for informasjonsutveksling [15], nemlig meldingsbasert, replikasjonsbasert og query-basert. Nedenfor gir vi en nærmere beskrivelse av disse tre mekanismene, før vi presenterer en foreløpig vurdering av hvordan de kan anvendes i et DG-miljø.

7.1.1 Meldingsbasert informasjonsutveksling

Meldingsbasert informasjonsutveksling innebærer at XML-dokumentene som utveksles er referansekomplette (referentially complete), med andre ord at det kun forekommer referanser til objekter som er inkludert i meldingen. Dette er den metoden som ble benyttet i demonstratoren under CWID 2006. Selve utvekslingen av meldinger kan være pull-basert, det vil si at mottakeren må sende forespørsler til avsenderen for å få nye meldinger, eller den kan være push-basert, hvor avsenderen automatisk sender ut notifications.

Den meldingsbaserte utvekslingsmekanismen kan så ytterligere klassifiseres etter hvordan man håndterer referanser til objekter:

Inline:
Dette innebærer at objekter som refereres innkapsles i objektet som refererer. En konsekvens av dette er at objekter som refereres av flere objekter da må replikeres (en kopi for hvert refererende objekt), noe som kan bidra til betydelig redundans. Fordelen med inline referanser er at man kan få en relativt oversiktlig struktur for mennesker, ved at alle refererte objekter er samlet i én ”pakke”. Dette er det imidlertid delte meninger om, da strukturene kan bli svært komplekse.

By reference:
Her benytter man en ”flat” struktur, og alle referanser til objekter realiseres ved hjelp av objektidentifikatorer (OID). På denne måten vil hvert objekt forekomme kun én gang i en melding, og man unngår redundans. Her er det imidlertid viktig at man har et gjennomført system for generering av OIDer. Disse må mappes direkte fra kjernen av et OO-system, slik at man er

sikret unike og meningsfylte OIDer, og det må også vurderes i hvilken grad de ulike sub-objektene trenger å utstyres med egne OIDer. Merk at disse kravene til OIDer også gjelder i mekanismene beskrevet under.

7.1.2 Replikasjonsbasert informasjonsutveksling

Her sørger man først for å utveksle et initialt informasjonssett, slik at avsender og mottaker er synkronisert i større eller mindre grad. Deretter er det kun oppdateringer som blir sendt, det vil si kun informasjon om endringen som har skjedd (for eksempel posisjon), samt OIDen for objektet det gjelder. Dermed er det heller ingen krav til at en melding skal være ”referentially complete”. Her benyttes normalt kun push-teknologi for å sende ut oppdateringer, og man kan enten sende ut oppdateringer hver gang endringer skjer, eller man kan samle opp og sende ut periodisk.

Replikasjonsbasert meldingsutveksling er den opprinnelige tanken bak JC3IEDM, og da som nevnt med sikte på relasjonsdatabaser. Samtidig er dette også hovedproblemet med OO-XML modellen, ved at hele databasestrukturen eksponeres. I et rent relasjonelt system vil dette fungere bra, siden det muliggjør en fullstendig replikasjon av databasen. I et objekt-orientert system er dette imidlertid uheldig, siden man opererer med helt andre strukturer, og typisk inneslutter objekter inne i andre objekter. Videre bør man unngå å måtte inkludere hele historien til objekter når man replikerer. Hensikten er å oppdatere om aktuell situasjon, og så må det være opp til de enkelte endesystemene om man ønsker å ta vare på historien.

7.1.3 Query-basert meldingsutveksling

Mens det for replikasjonsbasert meldingsutveksling er avsenderen som er den aktive part, er det her mottakeren som er aktiv, ettersom vi har en pull-basert meldingsutveksling. Mottakeren kan for eksempel be om informasjon om objekter i et gitt geografisk område, og så i etterkant be om mer informasjon om et av objektene som ble rapportert. Dette gjøres ved å referere til OIDen til det aktuelle objektet. Heller ikke query-basert meldingsutveksling har krav til at meldingene skal være referansekomplette. Dette åpner for at queryene kan være vilkårlig detaljerte, og at mottakeren dermed kan skreddersy spørringen til sine behov.

7.2 Anvendelser i taktiske nett

Prinsippene over er riktignok beskrevet som en del av MIP, men de er likevel uavhengige av datamodellene for informasjonsutveksling (C2IEDM og JC3IEDM). Så lenge det er mulig å entydig identifisere alle objekter, kan man benytte disse prinsippene i enhver modell for informasjonsutveksling. Vi vil derfor ta utgangspunkt i disse prinsippene når vi vurderer løsninger for informasjonsutveksling over DG.

Utgangspunktet vårt er altså nett med lav datarate, og til dels høy forsinkelse (inkludert snutid). Som vist i [16] er det en sammenheng mellom pakkestørrelse og dataraten som oppnås. Det viser seg at man må opp i pakker i størrelsesorden 2-5 kB før man får en akseptabel utnyttelse av dataraten (det vil si mer enn 50 % av maks oppnåelig datarate). Samtidig opererer vi med relativt lave datarater, gjerne under 2 kbit/s.

Dette betyr altså at vi på den ene side ønsker å sende så lite data som mulig, på grunn av lav datarate, men vi på den annen side ønsker å unngå å sende for små pakker, da dette gir dårlig utnyttelse av dataraten. Samtidig må man ta i betraktning at store pakker er mer sårbare for bitfeil, ved at mer data må retransmitteres. Dermed må man vurdere å benytte små pakker i nett med høy bitfeilrate. I tillegg ønsker vi å holde antallet retningsendringer i datastrømmen på et minimum, da snutiden ofte er høy.

7.2.1 Replikasjonsbasert informasjonsutveksling

Hvis vi nå ser på de ulike prinsippene for informasjonsutveksling beskrevet over, er det klart at replikasjonsbasert meldingsutveksling ikke er optimalt, spesielt ikke dersom man benytter hendelsesbasert meldingsutsending. Dette ville innebære at det sendes ut små meldinger hver gang informasjonen om et objekt oppdateres, noe som vil innebære svært dårlig utnyttelse av dataraten. Dersom man i stedet samler opp oppdateringer, og sender dem ut i ”batcher”, vil man oppnå bedre utnyttelse av dataraten, men man må da akseptere at informasjonen kanskje ikke er like aktuell.

I et nett med stor feilrate vil replikasjonsbasert meldingsutveksling dessuten øke sannsynligheten for at mottakere mister oppdateringer. Dette er spesielt problematisk dersom det går lang tid før informasjonen om et objekt endres på nytt. Man kan for eksempel tenke seg at et objekt reklassifiseres fra ”nøytralt” til ”fiendtlig”. Denne hendelsen sendes så ut som en oppdatering, men dersom en mottaker går glipp av denne oppdateringen, vil denne være uvitende om reklassifiseringen inntil det eventuelt publiseres ny informasjon om objektet.

7.2.2 Query-basert informasjonsutveksling

Query-basert meldingsutveksling innebærer at mottakeren aktivt ber om informasjon fra avsender. Dette kan være et helt situasjonsbilde, eller det kan være informasjon om enkeltobjekter. I noen taktiske nett vil det siste være spesielt uheldig, ettersom det både innebærer overføring av små informasjonsmengder og flere retningsendringer i datastrømmen. Den beste anvendelsen av query-basert utveksling vil trolig være ved å ”piggybacke” (hekte på) en forespørsel om et fullt situasjonsbilde på en forespørsel om et publish/subscribe abonnement. Dermed vil mottakeren, som svar på forespørselen om abonnement, motta et initialt situasjonsbilde.

7.2.3 Meldingsbasert informasjonsutveksling av MIP data

Når det gjelder meldingsbasert informasjonsutveksling[2], som blant annet ble benyttet under CWID 2006, er dette i utgangspunktet en lite effektiv måte å utveksle informasjon på, fordi den innebærer to former for redundans:

- Man sender ut informasjon om alle objekter, uavhengig av om informasjonen om dem har vært oppdaterer eller ikke
- Dersom man benytter inline referanser, blir gjerne mye av informasjonen sendt flere

[2] Vi antar at det benyttes push-basert meldingsutveksling

ganger innenfor samme melding

Til tross for den dårlige effektiviteten, kan dette prinsippet ha store fordeler i taktiske nett. For det første antar vi at det benyttes komprimering for å redusere størrelsen på meldingene. Fra CWID 2006 så vi at meldinger fort kunne komme opp i størrelser på 500kB (med inline referanser), men samtidig viser resultatene fra kapittel 4 at XML-baserte formater er svært komprimeringsvennlige, og en melding på 500kB reduseres gjerne ned til 10-15kB ved hjelp av generelle komprimeringsmetoder som GZIP.

I nett med høy feilrate kan dessuten redundansen i meldingene innebære en stor fordel. Dersom man skulle gå glipp av en oppdatering som inneholder for eksempel en reklassifisering som beskrevet i eksemplet over, vil dette ha liten betydning, ettersom man vil motta den samme informasjonen i neste melding. Dermed reduserer man også behovet for sende forespørsler fra mottakerne til avsenderen for å be om manglende informasjon, som i sin tur reduserer behovet for retningsendringer i datastrømmen. På den annen side vil det føre til noe økt belastning på mottakerne, ettersom disse må sjekke hvert enkelt mottatt objekt mot eksisterende objekter for å se om det er foretatt oppdateringer.

Hvorvidt man skal benytte ”inline” eller ”by reference” modellen er det vanskelig å gi noen generelle anbefalinger om. Dersom man benytter ”inline”-modellen vil redundansen øke ytterligere, men som nevnt i kapittel 6.1 er det mulig å redusere dette problemet gjennom å optimalisere XML-representasjonen. Ved å benytte ”by reference”-modellen reduseres redundansen, men samtidig må alle objekttyper ha en egen OID, og det stilles dermed større krav til generering og håndtering av slike OIDer.

En ytterligere fordel med meldingsbasert informasjonsutveksling er at det ikke er behov for å holde på tilstand for mottakerne. All informasjon sendes til alle mottakere, og så er det opp til den enkelte mottaker å gjøre bruk av den mottatte informasjonen. Dermed ligger også forholdene godt til rette for bred utnyttelse av multicast. Her kan man også tenke seg bruken av proxies, som filtrerer informasjonen for enkelte mottakere, og eventuelt holder på tilstand for disse.

Ulempen med bruk av meldingsbasert utveksling er at utsendingshyppigheten nødvendigvis vil måtte begrenses, for å unngå overbelastning av nettet. Dermed vil man måtte leve med en viss forsinkelse fra informasjonen om et objekt faktisk endres og til mottakerne blir oppdatert. Spesielt hvis det er svært stor aktivitet i et situasjonsbilde, det vil si man har svært mange og hyppige oppdateringer av objektene, eller man har behov for å motta oppdateringer så raskt som overhodet mulig, kan det være aktuelt å gå over til replikasjonsbasert oppdatering.

7.2.4 Kombinasjoner

Det kan også være aktuelt å se på kombinasjoner av de tre prinsippene. Det mest nærliggende er antakelig en kombinasjon av meldings- og replikasjonsbasert informasjonsutveksling. I praksis vil dette si at man benytter meldingsbasert utveksling, men at man fjerner kravet om at meldingene skal være referentially complete. Dermed kan man sende meldinger som ikke nødvendigvis

omfatter alle objekter i situasjonsbildet, og man trenger heller ikke sende all informasjon om alle objekter. På den måten kan man for eksempel sende ut meldinger som inneholder alle objekter hvor informasjonen er endret, og i tillegg objekter som er definert som ”viktige” (for eksempel alle fiendtlige objekter), selv om disse ikke har fått endret informasjon. Her bør det imidlertid undersøkes hvor mye man sparer av plass ved en slik fremgangsmåte, sammenliknet med standard meldingsbasert utveksling.

Dersom man har tilgang til nett med høyere datarate og lavere forsinkelse er det også aktuelt med en kombinasjon av query- og replikasjonsbasert informasjonsutveksling. Dermed kan mottakeren benytte query-basert utveksling for å hente ned et initialt situasjonsbilde, og så basere seg på replikasjonsbaserte oppdateringer videre. Skulle det være ønske om mer informasjon om et objekt, kan mottakeren igjen benytte en query for å hente denne informasjonen.

7.3 Informasjonsutveksling basert på synkronisering

Synkronisering basert på deltakoding er en metode for synkronisering og speiling av filsystemer med minimal datautveksling. En kjent implementasjon av dette prinsippet er *rsync* [17]. Selve prinsippet går ut på å dele opp dataene som skal synkroniseres i en serie like store blokker (”chunks”), og så beregne en sjekksum for hver blokk. Man utveksler så sjekksummer, og beregner hvilke blokker som er ulike. Disse blokkene blir så utvekslet, sammen med informasjon om hvordan de skal integreres med de eksisterende blokkene.

I XML-sammenheng er imidlertid problemet med slik deltakoding at man beregner sjekksummen ”binært”, det vil si man beregner sjekksummen ut fra karakterene i blokkene. Dermed er en endring av en enkelt karakter nok til at sjekksummen endres, og den tilsvarende blokken må utveksles. Reglene for XML-koding er på sin side relativt fleksible, slik at samme dokumentstruktur og informasjon kan representeres av ulike XML-dokumenter. En konsekvens av dette er at ulike serialiseringsverktøy genererer ulike versjoner av et og samme dokument (dette kan for eksempel gjelde ting som bruk av blanke tegn og linjeskift, eller hvor namespaces deklareres). Dermed vil logisk ekvivalente dokumenter få ulik sjekksum, og følgelig bli oppfattet som ulike, slik at data utveksles.

Før man kan beregne sjekksum for et XML-dokument må man derfor få det over på en kanonisk form, dokumentet må ”kanonikaliseres” (canonicalization). Dette innebærer at dokumentet konverteres til en standardisert form, med hensyn til koding, bruk av linjeskift, og så videre. Her pågår det standardiseringsarbeid innen W3C [18], i første rekke med fokus på digitale signaturer som anvendelsesområde. Det gjenstår derfor å undersøke hvordan en slik kanonisk form vil fungere i sammenheng med synkronisering.

Generelt bør også en eventuell bruk av synkronisering vurderes opp mot det aktuelle mønsteret for informasjonsutveksling: Så lenge man er ”online” er det antakelig enklere at den part som har fått oppdatert informasjon om et objekt sender ut dette til mottakerne. Hvis man derimot har vært ”offline” over en periode, og begge parter har fått oppdatert informasjon uavhengig av hverandre, kan synkronisering være en effektiv måte å sørge for at begge parter blir oppdatert på.

8 Bruk av proxies

I dette kapittelet vil vi diskutere hvordan proxies kan benyttes til å optimalisere bruken av Web Services over DG, og foreslå noen teknikker som kan være aktuelle i et slikt scenario.

8.1 Introduksjon

En proxy er en enhet som fungerer som et mellomledd mellom to parter som kommuniserer over et nettverk. En slik proxy kan tilby en rekke funksjoner som blant annet kan bidra til å redusere overhead, øke sikkerhet, gi høyere tilgjengelighet og senke forsinkelsen i forbindelse med kommunikasjon over nettverk.

Tre av de vanligste proxy funksjonene er:

- forbindelseshåndtering
- mellomlagring
- filtrering og brannmursfunksjoner.

Proxies er i utstrakt bruk i dag, for eksempel webproxier som Squid [28] som mellomlagrer websider. Det finnes i skrivende stund ingen proxyløsninger for Web Services, men vi presenterer her hvordan kjente teknikker kan tilpasses til bruk med Web Services.

En proxy kjennetegnes ved at den opererer på applikasjonslaget, som er lag 7 i OSI-modellen. Dette medfører at en proxy har tilgang til mer informasjon enn en ruter eller en brannmur som opererer på lavere lag i OSI-modellen, og kan dermed benytte mer avanserte og krevende teknikker for eksempel i forbindelse med filtrering.

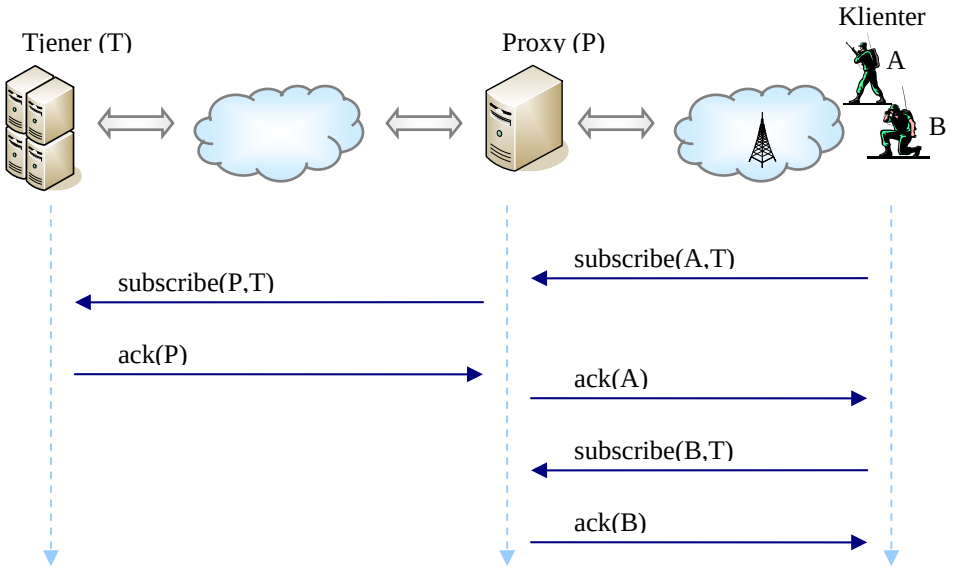
En proxyløsning kan implementeres på to måter: En transparent proxy er usynlig for klienten og kan således benyttes uten å måtte endre på eksisterende klient- og tjenerimplementasjoner. En tradisjonell proxy er derimot ikke usynlig for klienten, og klienten må konfigureres til eksplisitt å kommunisere via proxyen.

8.2 Forbindelseshåndtering

Proxies kan benyttes til håndtering av mange ulike typer nettverksforbindelser, et eksempel på et vanlig bruksområde er deling av en nettverksforbindelse mellom mange brukere.

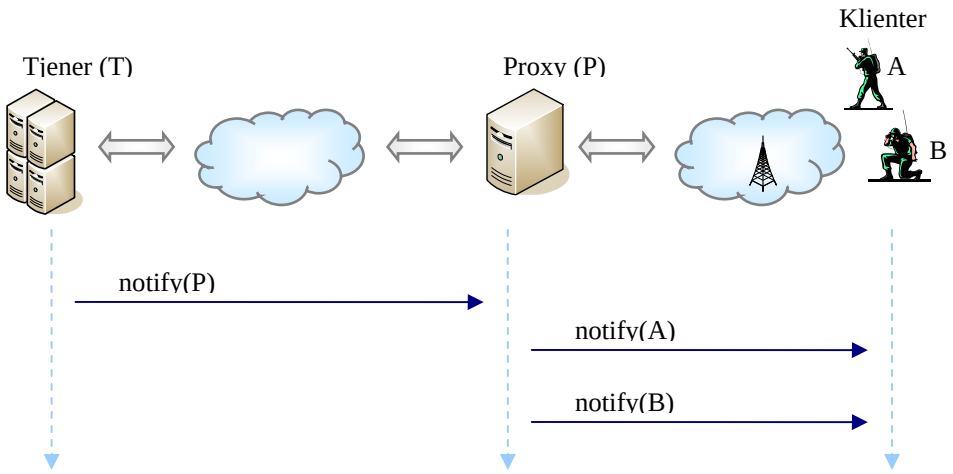
Abonnementstjenester, som beskrevet i kapittel 7, er en svært aktuell kommunikasjonsform når Web Services skal benyttes i nettverk med begrensede ressurser. I sin enkleste form vil slike abonnementstjenester føre til at mange kopier av samme melding sendes ut over det samme nettverket. I et DG bør dette unngås i størst mulig grad ved at bruk av multicast på delte kanaler benyttes der dette er tilgjengelig. Proxies kan benyttes til å innføre multicast over slike kanaler selv om abonnementstjenesten ikke støtter dette selv.

Vi skal nå presentere et eksempel på hvordan en proxy kan benyttes til å effektivisere abonnementstjenester. En abonnementstjeneste har en tjener som håndterer abonnementer (subscriptions) og sender ut oppdateringer (notifications) til abonnentene sine. Når en klient ønsker å benytte seg av en slik tjeneste må den først tegne et abonnement. Dette gjøres ved å sende en ”subscribe”-melding til tjeneren, som så bekrefter at denne er mottatt og at abonnementet er opprettet. Figurene nedenfor viser hvordan en transparent proxy kan benyttes for å effektivisere abonnementstjenester. Figur 8.1 beskriver oppstartsfasen der to klienter ønsker å tegne et abonnement hver på samme tjeneste. Først sender klient A en ”subscribe”-melding til tjener T. Mellom klient og tjener finnes en transparent proxy P som fanger opp denne meldingen. Ettersom dette er første gang P ser en slik melding til T lager den en egen ”subscribe”-melding som den sender til T. T bekrefter til P at et abonnement er opprettet. Nå har altså T fått P som abonnent, og P vet at oppdateringer fra T skal videresendes til A. På dette punktet sender P en bekreftelse til A på at abonnementet er opprettet, men utgir seg for å være T. Senere kan flere klienter komme til, i eksempelet under vil en klient B tegne samme abonnement som A har. B sender derfor en ”subscribe”-melding til T som fanges opp av P. Ettersom P allerede har tegnet det samme abonnementet på vegne av A, oppretter den ikke et nytt abonnement hos T. Den lagrer derimot lokalt at B er interessert i de samme oppdateringene som A, og sender en melding til B om at abonnementet er tegnet hos T.

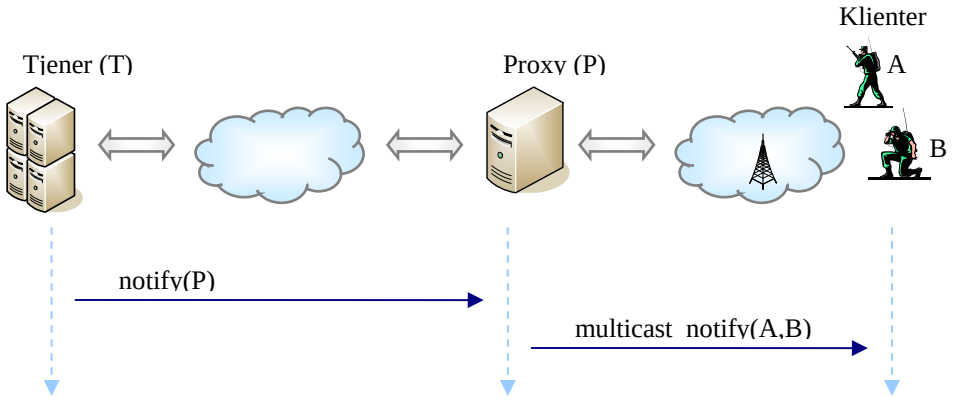


Figur 8.1 Tegning av abonnement via transparent proxy

Når abonnementet er tegnet vil videre kommunikasjon foregå asynkront ved at tjeneren sender ut en oppdatering når den har noe ny informasjon. Figur 8.2 illustrerer nettopp dette: T har ny informasjon til abonnentene sine og sender ut en ”notification”-melding. P har tegnet et abonnement på vegne av A og B, og mottar denne meldingen. P vil så sjekke sin interne abonnementsliste og finne ut at en og samme melding må videresendes til både A og B. Hvordan P gjør dette vil avhenge av transmisjonsmediet mellom P og klientene. Dersom mediet ikke støtter multicast vil P måtte sende en ”notification”-melding til hver klient, som vist i Figur 8.2. Unicast-distribusjon av ”notification”-meldinger er ikke ønskelig dersom multicast er tilgjengelig. Multicast vil redusere den totale trafikkmengden. Figur 8.3 viser hvordan proxyen kan lage én multicast-melding til begge sine abonnenter.



Figur 8.2 Abonnement via proxy, unicast



Figur 8.3 Abonnement via proxy, multicast

Det finnes både fordeler og ulemper ved å benytte en transparent proxy på denne måten. En viktig fordel er at man sparer båndbredde i nettverket som ligger mellom tjener og proxy, samt at proxyen kan tilpasse meldingene til transmisjonsmediet mellom seg og sine klienter ved for eksempel å bytte fra unicast til multicast. Dette vil medføre en kraftig reduksjon i båndbreddebruk ettersom man da får totalt én multicast-melding i stedet for en unicast-melding til hver klient. Ytterligere reduksjon i båndbredde kan oppnås ved filtrering, dette diskuteres videre nedenfor.

En ulempe ved å innføre en proxy i nettverket er at man da trenger å holde oppdatert tilstand i proxyen også, og ikke bare i tjeneren. Dette innfører et nytt punkt i systemet som er sårbart og kan feile. Dersom en proxy mister tilstandsinformasjonen sin vil den ikke vite hva den skal gjøre med meldinger den mottar fra tjenere. Man må derfor se nærmere på ulike løsninger for oppdatering av abonnementer for å overkomme dette problemet. Et annet problem som oppstår når man innfører en transparent proxy er at man mister ende-til-ende-bekreftelser (ack). Ettersom proxyen fanger opp kommunikasjon og utgir seg for å være klient overfor tjeneren og tjener overfor klienten så vil ikke tjeneren vite hvor mange abonnenter den faktisk har, og den vil heller ikke vite om alle disse har mottatt meldingene. Dersom man har behov for ende-til-ende-bekreftelser må man se på muligheten for å lage en hybridløsning der proxyen kan effektivisere

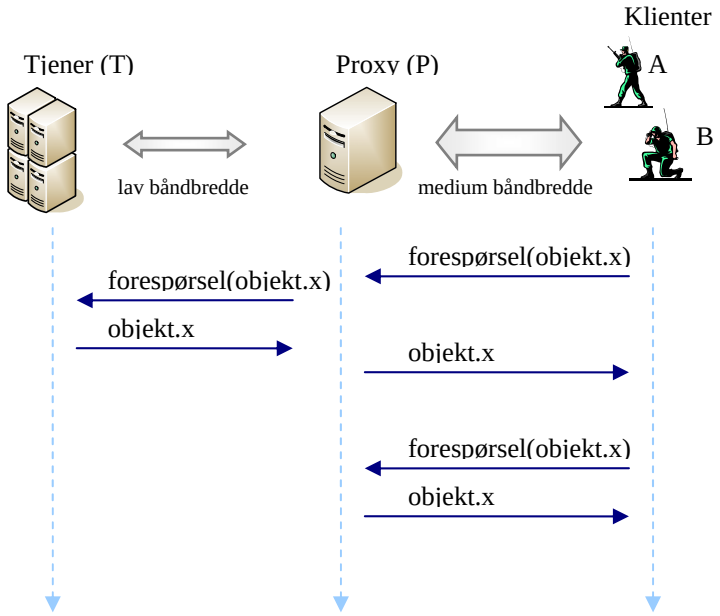
leveringen av ”notification”-meldinger, mens bekreftelser sendes på en måte som identifiserer alle mottakere.

8.3 Mellomlagring

En teknikk som er mye brukt for å bedre responstiden for klienter, samtidig som man minsker belastningen på opprinnelsestjeneren og nettverket den er tilknyttet, er mellomlagring (caching). Når man mellomlagrer lager man en kopi av data nær klienten etter at en forespørsel er gjort. Eventuelle senere forespørsler etter samme data vil da hentes fra mellomlageret og ikke opprinnelsesstedet. Mellomlagring foretas altså på bakgrunn av klienters bruk, og er altså en teknikk som er komplementær til replikering. Replikering fører også til at det lages en kopi av data, men er en prosess drevet fra tjenersiden der tjeneren vil ha ansvar for å holde orden på replikaer og oppdatere dem når det er nødvendig. Proxyer benyttes ofte til mellomlagring. Dette er spesielt viktig i DG, der tilgjengelig datarate ofte kan være veldig liten. Vi skal nå se nærmere på et eksempel som viser potensielle besparelser ved å benytte denne teknikken.

Figur 8.4 illustrerer klienter i et nettverk, der en proxy er koplet opp mellom nettverket klientene befinner seg på og resten av verden. All kommunikasjon til og fra det nettverket som klientene befinner seg på vil måtte gå gjennom denne proxyen. P opptrer altså som en gateway for disse klientene. I tillegg til tradisjonelle gatewayoppgaver som for eksempel adgangskontroll, så opptrer P her som et mellomlager. Dersom klient A ber om ”objekt.x” fra tjeneren T, så vil denne forespørselen nødvendigvis måtte gå gjennom P først. P inspiserer forespørselen, og finner ut at dette er en forespørsel til et objekt den ikke kjenner til fra før. P sender så forespørselen videre til T, og opptrer som klient overfor denne tjeneren. T finner fram ”objekt.x” og sender det til P. P mottar objektet, mellomlagrer det, og videresender det til A som sendte forespørselen. Hvis nå ennå en klient på et senere tidspunkt er interessert i å få tak i ”objekt.x” og sender en forespørsel, så vil denne kunne betjenes direkte fra P uten å måtte gå via T. Dette sparer da både tiden det ville ta å få overført objektet fra T til P, samt de datarate ressursene objektet ville ha trengt for denne overføringen.

Bruk av mellomlagring er spesielt nyttig dersom kapasiteten til nettverket på tjenersiden er som i Figur 8.4, altså at flaskehalsen ligger i nettverket mellom tjener og proxy. Da vil all trafikk som P kan håndtere uten å måtte kommunisere med T, spare datarate ressurser. I et taktisk nett vil ofte siste ledd, altså nettverket mellom proxy og klient, også ha lav datarate. Bruken av mellomlagring vil da sørge for at forespørsel og svar kun må gå over én lavkapasitetslink, noe som reduserer forsinkelsen mellom sendt forespørsel og mottatt svar i tillegg til båndbreddebesparelsen. Mellomlagring bidrar også til å gjøre systemet mer robust ved å øke tilgjengeligheten; objektet finnes i proxyen selv om den opprinnelige kilden blir utilgjengelig.



Figur 8.4 Mellomlagring

Det er imidlertid også en del utfordringer forbundet med mellomlagring av objekter. En proxy må kjenne og forstå de objektene den skal mellomlagre, slik at den håndterer objektene på en fornuftig måte. Dersom et objekt har begrenset levetid må proxyen være oppmerksom på dette, og sørge for å hente en ny versjon fra tjeneren dersom det kommer en forespørsel fra en klient etter at den mellomlagrede kopiens levetid er over. Mellomlagring fungerer ikke effektivt på objekter som endres ofte. En proxy vil dessuten ikke ha ubegrenset lagringsplass, så den vil ofte måtte foreta et valg mellom hvilke objekter som skal få være i mellomlageret og hvilke som skal kastes ut for å gi plass til nye. Det finnes ulike algoritmer for å gjøre dette, for eksempel benyttes statistiske algoritmer ofte i proxy-sammenheng.

8.4 Filtrering

Når man opererer over nettverk med begrenset kapasitet er det spesielt viktig at man ikke oversvømmer nettverket med mer informasjon enn det som kan håndteres. Samtidig må man tilstrebe at klientene får all den viktige og relevante informasjonen de har behov for.
For å få til dette trenger man filtrering av informasjon mellom nettverk. Filtrering kan deles inn i tre hovedtyper basert på hva det er som filtreres:

1. Filtrering på forbindelsesnivå, det vil si om man får lov til å opprette en forbindelse eller ikke. Dette kan benyttes både i sikkerhetssammenheng og for å prioritere ressursbruk.
2. Filtrering av hele meldinger baseres på kriterier som for eksempel meldingens prioritet, relevans og gradering. Meldinger med høy prioritet må slippes gjennom før meldinger med lavere prioritet.
3. Filtrering av deler av meldinger basert på en vurdering av meldingens innhold.

En proxy kan foreta alle tre typer filtrering. Type 1 og til dels type 2 filtrering kan foretas på nettverkslaget av en ruter, mens mer avanserte former for filtrering krever løsninger på applikasjonsnivå.

9 Konklusjon

Vi har i denne rapporten undersøkt ulike aspekter ved det å benytte SOA i NbF. Ettersom Web Services basert på XML og SOAP er den vanligste implementasjonen av SOA i dag, har vi valgt å fokusere på dette. Fokus er på taktiske nett med liten tilgjengelig datarate, såkalte Disadvantaged Grids (DG). Vi har undersøkt ulike måter å redusere XML-overhead på, og vi har simulert en Web Service over XOmail – en implementasjon av STANAG 4406. Vi har også kommet med forslag til ytterligere måter man kan forbedre systemets ytelse på, blant annet ved å optimalisere meldingsrepresentasjon og -innhold, samt ved å benytte proxies.

Undersøkelsene våre viser at man må benytte komprimering for å kunne kjøre Web Services over DG. ZLIB en standardisert komprimeringsalgoritme som komprimerer bra og raskt. Dersom man skal velge en generell algoritme som fungerer godt, så er dette valget. Samtidig ser vi at XML-spesifikke metoder ofte gir bedre komprimering, men at disse også har betydelig lenger prosesseringstid. Våre målinger viser at dersom man sender XML-dokumenter mindre enn 30 KB, så gir den proprietære Efficient XML best resultat, mens XMILL er best på større XML dokumenter.

XOmail er en implementasjon av STANAG 4406 fra Thales. XOmail støtter flere ulike kommunikasjonsprofiler, deriblant TMI og DMP som er spesielt optimalisert for kommunikasjon i taktiske nettverk. Vi har sammenliknet bruken av disse to kommunikasjonsprofilene over en linksimulator der vi har benyttet datarater som er typiske for DG. Våre undersøkelser viser at DMP er det mest effektive alternativet, og at denne profilen bør benyttes dersom det er mulig. DMP støtter ikke full meldingsfunksjonalitet som spesifisert i STANAG 4406, og i tilfeller hvor slik funksjonalitet er påkrevd er TMI et godt alternativ.

Eksperimentene indikerer at det vil være mulig å benytte Web Services også over DG, forutsatt at man komprimerer data og benytter en effektiv kommunikasjonsprofil. Dersom man benytter XOmail som bærer, så har både TMI- og DMP-profilen ZLIB komprimering innebygget, noe som vil være velegnet for kommunikasjon over såvel taktiske som strategiske nettverk. Forutsetningen for at dette skal virke vil være at hyppigheten på meldingene som sendes ut ikke er så stor at den overskrider nettets kapasitet selv ved bruk av komprimering. Man kan i slike tilfeller tenkte seg å benytte ulike teknikker for å optimalisere selve meldingsutvekslingen. Vi har presentert følgende teknikker:

1. Optimalisering av informasjonsrepresentasjon
2. Optimalisering av informasjonsutveksling
3. Bruk av proxies

Disse teknikkene vil føre til enda bedre utnyttelse av den kapasiteten man har til rådighet i kommunikasjonssystemene. Det finnes i dag ingen kjente implementasjoner av de nevnte teknikkene som fungerer med Web Services, XML og SOAP. Videre forskning er nødvendig for å undersøke disse løsningene i detalj.

Referanser

[1] Efficient XML Interchange Working Group. http://www.w3.org/XML/EXI /

[2] NIST Net, a Linux Network Emulation Tool. http://www-x.antd.nist.gov/nistnet/

[3] Sun Developer Network – Fast Infoset. http://java.sun.com/developer/technicalArticles/xml/fastinfoset/

[4] Fast Infoset specification. ITU-T Rec. X.891 | ISO/IEC 24824-1. http://www.itu.int/rec/T-REC-X.891/e

[5] ASN.1 information site. http://asn1.elibel.tm.fr/

[6] Fast Web Services specification. ITU-T Rec. X.892 | ISO/IEC 24824-2. http://www.itu.int/rec/T-REC-X.892/e

[7] GZIP home page. http://www.gzip.org/

[8] ZLIB home page. http://www.zlib.net/

[9] XMILL open source project. http://sourceforge.net/projects/xmill

[10] Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER). http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf

[11] Information technology - ASN.1 encoding rules: XML encoding rules (XER). http://www.itu.int/ITU-T/studygroups/com17/languages/X.693-0112.pdf

[12] Information technology - ASN.1 encoding rules: Specification of Packed Encoding Rules (PER). http://www.itu.int/ITU-T/studygroups/com17/languages/X.691-0207.pdf

[13] Generic String Encoding Rules (GSER) for ASN.1 Types. RFC3641. http://tools.ietf.org/html/rfc3641

[14] Simple Object Access Protocol (SOAP) 1.1 Specification. World Wide Web Consortium (W3C). http://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383526

[15] Gerz Michael, An Object-Oriented XML Schema for the MIP Joint Command, Control, and Consultation Information Exchange Data Model, in 2006 Command and Control Research and Technology Symposium, (CCRTS), San Diego, CA, USA, June 2006

[16] Jodalen Vivianne, Solberg Bjørn, Grønnerud Ove, Leere Anton, Performance testing of STANAG 4406 (Military messaging using IP over HF), FFI-rapport 2005/01183, 2005

[17] Rsync home page, http://rsync.samba.org/

[18] W3C, Canonical XML Version 1.0, http://www.w3.org/TR/xml-c14n

[19] SOAP messages with attachments, W3C note. http://www.w3.org/TR/SOAP-attachments

[20] SOAP Message Transmission Optimization Mechanism, W3C Recommendation. http://www.w3.org/TR/soap12-mtom/

[21] XML-binary Optimized Packaging, W3C Recommendation. http://www.w3.org/TR/xop10/

[22] Efficient XML (EFX), AgileDelta. http://www.agiledelta.com/product_efx.html

[23] Rose K, Lund K, Sletten G (2006): Suggested improvements to the MIP Web Services/Object-Oriented XML Information Exchange Data Model, FFI/NOTAT-2006/03599, Forsvarets forskningsinstitutt

[24] Rose K (2006): Binærkoding av XML-dokumenter – En innledende undersøkelse, FFI/NOTAT-2006/02474, Forsvarets forskningsinstitutt

[25] STANAG 5527 – STUDY DRAFT 1 – NATO Friendly Force Information Standard for Interoperability of Force Tracking Systems, AC/322(SC/5)N(2006)0027

[26] Mevassvik O M, Hafnor H, Hanssen B J, Leere A B, Rose K, Sanden H, Urdahl M, Viken K O, Bråthen K (2003): Demonstrator for bildeoppbygging, FFI/RAPPORT-2003/02748, Forsvarets forskningsinstitutt

[27] Introduksjon til Nettverksbasert Forsvar. Forsvarets stabsskole 2001. Militærteoretisk skriftserie nr.1

[28] Squid Web Proxy Cache. http://www.squid-cache.org

[29] INI materiell prosjekt 8002: Gjennomgående militær meldingstjeneste

[30] NATO Consultation, Command and Control Agency, NATO Network Enabled Capability Feasibility Study, Version 2.0, Oktober 2005

Forkortelser

AFTS	Afghanistan Force Tracking System
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CER	Canonical Encoding Rules
C2IEDM	Command and Control Information Exchange Data Model
CWID	Coalition Warrior Interoperability Demonstration
DMP	Direct Messaging Profile
DG	Disadvantaged Grids
DER	Distinguished Encoding Rules
EFX	Efficient XML
XML	Extensible Markup Language
FI	Fast Infoset
FTP	File Transfer Protocol
GSER	Generic String Encoding Rules
GPS	Global Positioning System
HQ	Headquarters
HF	High Frequency
HB-UHF	High-band Ultra High Frequency
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
INI	Informasjonsinfrastruktur
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union- Telecommunication Standardization Sector
IETF	Internet Engineering Task Force
JC3IEDM	Joint Command, Control and Consultation Information Exchange Data Model
LFR	Lett Feltradio
MIF	Meldingstjenesten i Forsvaret
MTOM	Message Transmission Optimization Mechanism
MMHS	Military Message Handling System
MIP	Multilateral Interoperability Programme
MIME	Multipurpose Internet Mail Extensions
MRR	Multirolleradio
NFFI	NATO Friendly Force Information
NNEC	NATO Network Enabled Capability
NbF	Nettverksbasert Forsvar
OID	Object ID

OO	Object Oriented
OSI	Open Systems Interconnect
PER	Packed Encoding Rules
PRR	Personal Role Radio
RTT	Round-trip time
SOA	Service-oriented Architecture
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
STANAG	Standardization Agreement
SNR	Subnetwork Relay
TMI	Tactical Messaging Interface
WSDL	Web Services Description Language
W3C	World Wide Web Consortium
XER	XML Encoding Rules
XSD	XML Schema Definition