FFI RAPPORT

ARKITEKTURAR OG STANDARDAR FOR DRIFT OG STYRING AV KOMMUNIKASJONSNETT

WINJUM Eli

FFI/RAPPORT-2001/04331

FFIE/794/110

Godkjent
Kjeller 5 september 2001

Torleiv Maseng
Forskningssjef

ARKITEKTURAR OG STANDARDAR FOR DRIFT OG STYRING AV KOMMUNIKASJONSNETT

WINJUM Eli

FFI/RAPPORT-2001/04331

FORSVARETS FORSKNINGSINSTITUTT
Norwegian Defence Research Establishment
Postboks 25, 2027 Kjeller, Norge

FORSVARETS FORSKNINGSINSTITUTT (FFI)
Norwegian Defence Research Establishment

P O BOX 25
NO-2027 KJELLER, NORWAY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

REPORT DOCUMENTATION PAGE

1) PUBL/REPORT NUMBER	2) SECURITY CLASSIFICATION	3) NUMBER OF PAGES
FFI/RAPPORT-2001/04331	UNCLASSIFIED	131
1a) PROJECT REFERENCE	**2a) DECLASSIFICATION/DOWNGRADING SCHEDULE**	
FFIE/794/110	-	

4) TITLE

ARKITEKTURAR OG STANDARDAR FOR DRIFT OG STYRING AV KOMMUNIKASJONSNETT

ARCHITECTURES AND STANDARDS FOR NETWORK MANAGEMENT

5) NAMES OF AUTHOR(S) IN FULL (surname first)

WINJUM Eli

6) DISTRIBUTION STATEMENT

Approved for public release. Distribution unlimited. (Offentlig tilgjengelig)

7) INDEXING TERMS

IN ENGLISH:	IN NORWEGIAN:
a) Network Management	a) Drift og styring av nett
b) Telecommunication Management Network	b) Telecommunication Management Network
c) Simple Network Management Protocol	c) Simple Network Management Protocol
d) Network Management Information	d) Informasjon for drift og styring av nett
e) Network Management Protocol	e) Protokoll for drift og styring av nett

THESAURUS REFERENCE:

8) ABSTRACT

The report reviews present architectures and standards for network management. It especially investigates the context and structure of management information as well as the information exchange in network management systems built upon the two dominating families of standards.

An overview on future proposals is given. Partly in contrast to present architectures, these concepts emphasize distributed network management. Important trends in this development are presented.

9) DATE	AUTHORIZED BY	POSITION
	This page only	
5 September 2001	Torleiv Maseng	Director of Research

ISBN-82-464-0563-2

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

INNHOLD

Side

ARKITEKTURAR OG STANDARDAR FOR DRIFT OG STYRING AV KOMMUNIKASJONSNETT

SAMANDRAG

Allment

Standardar for drift og styring av kommunikasjonsnett har tradisjonelt vore utvikla av og for to ulike miljø som har vektlagt ulike aspekt:

- Aktørar innanfor *Open System Interconnection* (OSI)-baserte telenett/transportnett har utvikla arkitekturar og standarder i regi av *International Standardization Organisation* (ISO) og *International Telecommunication Union - Telecommunication Standardization Sector* (ITU-T). Dette har resultert i *Telecommunication Management Network* (TMN) som i dag vert nytta av teleoperatørar. TMN-spesifikasjonane er under kontinuerleg utvikling, og har i seinere tid vorte påverka av og teke opp i seg nye konsept for objektorientert distribuert dataprosessering.
- Aktørar innanfor *Internet Protocol* (IP)-baserte datanett har utvikla arkitekturar og standardar i regi av *Internet Engineering Task Force* (IETF). Dette har resultert i system basert på *Simple Network Management Protocol* (SNMP). Desse standardane var opprinneleg meint å skulla vara ei kort stund, men dei har vorte svært utbreidde i IP-baserte nett. Dei er òg under stadig utvikling, og SNMP finns no i tre versjonar. Det er ingen grunnleggjande skilnad på dei to første versjonane. Versjon 3 introduserer først og fremst løysingar for datasikring. Dei tre versjonane er ikkje kompatible. Dei fleste system er framleis basert på versjon 1.

Tidlegare skilde desse to nettypane seg sterkt med omsyn til talet på nodar, kompleksitet, utstyrtypar, krav til tenestekvalitet, feil- og alarmhandtering, endringsfrekvens osv. Dei to konsepta for drift og styring speglar dette klart. Dei siste åra har telenett og datanett i stor grad konvergert:

- Ein finn *både* tradisjonelle datanett- *og* tradisjonelle tele/transport*nettelement* i eitt og same hybride *nett*.
- Ein finn *både* tradisjonelle datanett- *og* tradisjonell tele/transportnett*funksjonalitet* i eitt og same hybride *nettelement*.

Dette medfører utfordringar til drift- og styringssystema:

- Krava til tenestekvalitet vert meir like.
- Kommunikasjonsnetta er hybride med omsyn til dataprosesserings- og transport-funksjonalitet, og dei inneheld både OSI- og IP-baserte nettelement.
- Kommunikasjonsnettet kan som eit heile sjåast som eit stort distribuert dataprosesserings-system.

Nyare arkitekturar for drift og styring av kommunikasjonsnett speglar desse momenta. Ein

prøver i tillegg å finna kortsiktige løysinger som kan integrera dei gamle standardane.

Sjølv om omgjevnader og krav har vore ulike, finn ein likevel dei same grunnleggjande elementa i dei to drift- og styringskonsepta:

- *Agent*. Dette er ein applikasjon i eit kva som helst element i nettet, til dømes i en IP-rutar eller i ein SDH-multipleksar.
- *Manager*. Dette er ein applikasjon som styrer/overvakar nettelementa ved hjelp av agentapplikasjonane. I tradisjonelle drift- og styringssystem vil managerapplikasjonen vera sentralisert. I nyare konsept søkjer ein å distribuera managerfunksjonaliteten utover i nettet.
- *Informasjonsbase*. Desse er basert på standardiserte informasjonsmodellar. I ulik grad prøver modellane å avbilda sjølve nettelementa og relasjonane mellom dei. Både agent og manager vil ha tilgang til slike informasjonsbasar.
- *Kommunikasjonsnett*. Manager- og agentapplikasjonane utvekslar informasjon ved hjelp av eigne drift- og styringsprotokollar. Informasjonen som vert utveksla, kan grovt delast inn i to kategoriar:
  - *Operasjonar*. Som hovudregel er dette meldingar frå managerapplikasjonen om å oppretta, lesa, skriva over eller sletta data i informasjonsbasen til ein agent.
  - *Notifikasjonar*. Som hovudregel er dette meldingar fra ein agentapplikasjon om hendingar i domenet til agenten. Dette kan til dømes vera alarmer som varslar om feil i eit nettelement.
- *Brukargrensesnitt* for overvaking og manuell handtering.

Applikasjonsstruktur

ISO/ITU-T definerer fem overordna funksjonelle område for drift og styring av storskala kommunikasjonssystem. Desse områda ser ut til å vera ei allment godteken avgrensing av omgrepet *management*, når dette er knytt til kommunikasjonsnett. Dei fem områda er:

- *Feilhandtering* som omfattar deteksjon, isolasjon og korreksjon av unormale operasjonar.
- *Bruksregistrering* som omfattar identifikasjon av ressursbruk for mellom anna å kunna taksera denne.
- *Konfigurering* som først og fremst dreier seg om å gje ressursene parameterverdiar for å førebu, starta og terminera tenester.
- *Yting* som omfattar evaluering av ressursane og av effektiviteten i kommunikasjonen.
- *Datasikring* som omfattar allmenn støtte til eit eller fleire sett av gjeldande reglar (*policies*) for sikring av data og informasjon i trafikknettet; drift- og styringsinformasjonen inkludert.

Alle desse områda kan sjåast frå ulike abstraksjonsnivå: Eit nivå for drift og styring av det einskilde nettelementet, eit nivå for ulike fysiske og logiske samankoplingar (nett) av nettelement, eit tenestenivå og øvst eit nivå for heile verksemda. Grovt kan ein seia at noverande drift- og styringssystem dekkar element- og til dels nettnivået. Nyare arkitekturar fokuserer spesielt ytterlegare abstraksjon med sikte på applikasjonsutvikling på tenestenivået.

I tillegg til funksjonelle område og abstraksjonsnivå, har applikasjonsstrukturen endå ein dimensjon i og med at at han i stor grad er teknologispesifikk. TMN har ein svært generisk arkitektur som i utgangspunktet kan omfatta, og eventuelt detaljerast for, alle typar teknologiar og nett. Sjølve applikasjonane for drift og styring er likevel i praksis utvikla for å dekka spesifikke svitsje- og/eller transportteknologiar. Når både TMN- og SNMP-baserte drift- og styringsapplikasjonar vert nytta i same nett, vert biletet endå meir samansett. At dei ulike

leverandørane lagar spesifikke løysingar innanfor ein og same teknologi, gjer ikkje situasjonen betre.

Eventuell integrasjon av teknologispesifikke drift- og styringsapplikasjonar har både positive og negative konsekvensar. I kva grad integrasjon er ynskjeleg, avheng mellom anna av kompleksiteten i nettet og kva for tenester operatøren/leverandøren skal støtta/levera.
Det finns ikkje stadardar for å intgrera drift og styring av ulike teknologiar. Utviklinga i retning transportnett basert på "IP-over-WDM" vil kunna framtvinga slike standardar.

Funksjonsmodellar
TMN er logisk delt inn i funksjonsblokkar med funksjonalitet. Med utgangspunkt i dei fem funksjonelle områda inneheld spesifikasjonane ei lang rekkje standardiserte drift- og styringsfunksjonar. Mellom funksjonsblokkane vert det definert referansepunkt, som i sin tur vert implementert som grensesnitt for blokken. Funksjonsblokken for eit nettelement vil dermed få eitt eller fleire referansepunkt som definerer kva for funksjonar dette elementet støttar i dette referansepunktet, til dømes kva for operasjoner som kan utførast mot objekta i informasjonsbasen til elementet, eller kva for alarmar elementet gir ved feil.

SNMP-spesifikasjonane inneheld ikkje tilsvarande standardiserte funksjonar. Mykje tyder på at SNMP-applikasjonar i all hovudsak vert nytta for *overvaking* av nett og i liten grad til *styring* av det, i form av til dømes konfigureringsfunksjonar.

Informasjonsmodellar
For TMN er det definert ein generisk objektorientert informasjonsmodell som kan spesialiserast for å dekka dei ulike nett-teknologiane. Dei generiske objektklassane vert gruppert i såkalla *fragment* og skal innehalda informasjon på både element- og nettnivå. Objektklassane vert definert med eit sett attributtar som skildrar eigenskapane, kva for operasjonar som er tilletne, samt notifikasjonar og åtferd. Parametrar som skal overførast til kommunikasjonsprotokollane, er òg definert. Elles inneheld spesifikasjonane fasilitetar som er vanlege for objektorienterte modellar; innkapsling, spesialisering og arv.

Informasjonen vert strukturert i samsvar med tre trestrukturar:
- *Registreringstre* som er ein global katalogstruktur introdusert av ISO/ITU-T. Strukturen gir eit kvart objekt fra eit kvart domene ei unik global adresse.
- *Arvetre* som syner korleis dei ulike objektklassane er relatert til kvarandre gjennom arv av eigenskaper.
- *Informasjonstre for drift og styring* som syner relasjonar mellom objekt. Dette kan vera modellar av hierarki i det virkelege kommunikasjonsnettet. Treet vert òg nytta for namngiving.

Samanlikna med TMN har SNMP ein svært enkel informasjonsmodell. Modellen er ikkje objektorientert. Objekta vert gruppert i objektgrupper og skal innehalda informasjon på elementnivået. Det er spesifisert påbyggjingsmodular for nettnivået. Eit SNMP-objekt er ein variabel, og kan samanliknast med ein attributt i ein objektorientert (eller i ein tradisjonell relasjons-) modell. Objekta vert definert med namn, syntaks (datatype), tilgang, status og tekstleg skildring. Objekta har skalare verdiar, men kan i informasjonsbasane organiserast i todimensjonale tabellar, slik at ein til ein viss grad kan syna relasjonar mellom dei.

Informasjonen vert strukturert i samsvar med det tidlegare nemnte globale registreringstreet.

Sjølv om relativt få informasjonsbasar har status som *Internet Standard*, finns det eit stort tal leverandør-, utstyr- og teknologispesifikke informasjonsbasar. Ikkje sjeldan er desse semantisk inkompatible.

Kommunikasjonsmodellar
Dei funksjonelle TMN-blokkane vert implementert som fysiske blokkar, og som nemnt, vert dei funksjonelle referansepunkta mellom desse blokkane implementert som grensesnitt.Dei ulike grensesnitta definerer kva for protokollar som kan nyttast mot ein fysisk blokk. Innan det TMN-interne kommunikasjonsnettet vil linjene vera av ulik type. Nettet skal så langt mogleg vera i tråd med OSI referansemodell. For dei låge kommunikasjonalaga er det definert ei rekkje protokollprofilar som kan nyttast, avhengig av den fysiske konfigurasjonen av TMN-systemet. På det øvste laget er protokollvalet avhengig av tenestetype. For interaktive tenester skal protokollen *Common Management Information Protocol* (CMIP) nyttast for informasjons-utveksling mellom manager- og agentapplikasjonar. Det er òg definert protokollar for filoverførings- og katalogtenester. TMN tillet dessutan bruk av *Common Object Request Broker Architecture* (CORBA) for kommunikasjon mellom TMN-domene. Det vert arbeidd med spesifikasjonar for bruk av CORBA i intra-domene-kommunikasjon.

Samanlikna med CMIP er SNMP ein enkel protokoll med avgrensa funksjonalitet. For å utveksla SNMP-meldingar over IP, er det nok med ei tilknytingsfri datagramtjeneste. *User Datagram Protocol* (UDP) er tilrådd som transportprotokoll, men både tilknytingsorienterte og tilknytingsfrie OSI-protokollar kan nyttast.

Datasikring
Datasikring er eit av dei fem funksjonelle områda for drift og styring av kommunikasjonsnett. For det første skal datasikringstenestene i trafikknettet handteras ved til dømes å forvalta og konfigurera mekanismar for kryptering, autentisering og tilgangskontroll. Åtak på gjeldande reglar for datasikring skal detekterast og rapporterast. Krava til sikring kan variera innanfor ulike nett og ulike domene. For det andre skal informasjonsutvekslinga mellom manager- og agentapplikasjonane samt tilgangen til informasjonsbasane i sjølve drift- og styringssystemet sikrast.

For TMN er det definert ei rekkje funksjonar for å handtera datasikringa i det underliggjande trafikknettet. Som del av eit OSI-miljø, skal TMN sjølv vera verna av det omfattande spekteret av tenester og mekanismar som er spesifisert i sikringsarkitekturen for OSI. For kommunikasjon innanfor eit TMN-domene er desse tenestene valfrie, og det vil vera opp til den einskilde TMN-eigaren å definera sikringsreglar. For inter-domene-kommunikasjon er autentiserings- og tilgangskontrollmekanismar obligatoriske på applikasjonslaget.

På grunn av mangelfulle sikringstenester i eksisterande IP-nett og på grunn av at SNMP i relativt liten grad vert nytta til å setja parametrar, er det lite truleg at system basert på SNMP versjon 1 og 2 handterer sikringssmekanismar i underliggjande nett. Med unnatak av ein svært svak autentiseringsmekanisme, har versjon 1 korkje tenester eller mekanismar for sikring av drift- og styringsinformasjonen. Dette gjeld for informasjonsutvekslinga mellom manager og agent så vel som for tilgangen til informasjonsbasane. Ser ein bort frå moglege utanforliggjande

tiltak, er SNMP-informasjon i praksis berre verna av sikringsmekanismar som IP tilbyr. For SNMP versjon 3 vert derimot eit utval tenester og mekanismar for datasikring spesifisert:

- Ein allmenn sikringsmodell som tilbyr autentisering av datakjelde, kryptering og tidssynkronisering av SNMP-meldingar.
- Ein modell for tilgangskontroll mot SNMP-entitetane.

Det er i dag usikkert i kva grad ein vil oppgradera SNMP-system til versjon 3, då *IP security* (IPsec) for mange kan vera eit aktuelt alternativ.

Nyare arkitekturar og trendar

Store kommunikasjonsnett er "av natur" *distribuerte system*. I tråd med dette synet er det utvikla arkitekturar for distribuert prosessering så vel som for drift og styring av slike distribuerte system. Sidan dei underliggjande kommunikasjonsnetta er distribuerte system, vil ein òg måtta utvikla drift- og styringssystema som distribuerte system. Arkitekturane for distribuerte system er difor viktige ut frå to aspekt:

- Som arkitekturar for det underliggjande kommunikasjonsnettet/systemet
- Som arkitekturar for sjølve drift- og styringsnettet/systemet

Fellestrekk ved disse arkitekturane er:

- Dei er basert på objektorienterte konsept og på generiske informasjonsmodellar.
- Dei har som viktig målsetting å skaffa til vege ei plattform for utvikling, produksjon og drift/styring av allmenne tenester så vel som av telekommunikasjonstenester. Denne plattforma skal mellom anna kunna skjula kompleksiteten i det underliggjande kommunikasjonsnettet.

Drift- og styringskonsepta fokuserer i tillegg:

- Integrasjon av eksisterande drift- og styringsstandardar.
- Drift og styring av distribuerte tenester og applikasjonar.
- Handtering av mange og ulike domene.

Dei meir framtidige konsepta legg stor vekt på å gjera kommunikasjonsnetta proaktive, sjølvregulerande og sjølvstyrte. Område det vert forska mykje på i samband med dette, er: Objektorientert mellomvareteknologi, agentteknologi og teknologiar for aktive nett (programmerbare nett).

1 INNLEIING

1.1 Bakgrunn for og formål med rapporten

Prosjektet *Trender innen samfunn og kommunikasjonsteknologi* har hatt ein delstudie av system for drift og styring av kommunikasjonsnett. Rapporten skal dokumentera dette arbeidet som i hovudsak har vore ein litteraturstudie av arkitekturar og standardar. Rapporten skal vera med å leggja grunnlag for vidare arbeid på dette feltet i prosjektet. Dette gjeld mellom anna ein planlagt rapport om systeminformasjon i kommunikasjonsnett. Rapporten kan vidare sjåast i samanheng med (49).

Eit kommunikasjonsnett er sett saman av ulike typar transmisjonsmedie og ulike typar hardware/software frå ei rekkje leverandørar. Komponentane finns i ulike versjonar, med ulik funksjonalitet, ulik informasjonsrepresentasjon, ulike fysiske og funksjonelle grensesnitt og med ulike kommunikasjonsprotokollar. Innanfor eit større nett vil krav til yteevne, kvalitet og datasikring dessuten kunna variera sterkt.

Ei rekkje organisasjonar kan vera ansvarlege for ulike deler av eit kommunikasjonsnett og/eller dei ulike tenestene som vert tilbydd frå og via dette. Nye tenester impliserer vidare at ein kunde skal kunna handtera visse parametrar for eigen trafikk. Denne tendensen vil forsterka seg. Alt dette medfører at drift- og styringssystema ikkje berre skal handtera eit teknologisk komplekst område, men i aukande grad, òg eit administrativt og juridisk.

For å handtera store kommunikasjonsnett effektivt, vil nettoperatørar så vel som teneste-tilbydarar nytta ei rekkje informasjonssystem samt system for fjernstyring og overvaking av nett og tenester. Det finns standardiserte informasjonsmodellar, kommunikasjonsprotokollar og funksjonalitet som skal dekka drift- og styringsfunksjonen for heile eller deler av kommunika-sjonsnettet. Det internasjonale arbeidet med å utvikla og standardisera drift og styring av kommunikasjonsnett er svært omfattande.

Formålet med rapporten er å kasta lys på nokre sider ved systemkonsepta som vert nytta i drift og styring av kommunikasjonsnett. Ein har særleg vore interessert i å sjå på drift- og styringsinformasjonen og utvekslinga av denne. For å få eit best mogleg bilete av dette, har ein valt å skildra dei ulike konsepta frå fire synsvinklar:

- Funksjonalitet
- Informasjonsinnhald og informasjonsstruktur
- Kommunikasjonsprotokollar
- Datasikring

Rapporten skal ikkje vurdera kor vidt dei ulike konsepta er *gode*.

Ein har i denne studien valt å sjå på standardar og arkitekturar utarbeidd av store, og meir eller mindre uavhengige, internasjonale organisasjonar og fora. Fordelen med dette er at desse organisasjonane i stor grad tek utgangspunkt i ein operativ heilskap, og dei ulike arkitekturane og spesifikasjonane gir eit godt bilete av omfanget av, og kompleksiteten i, drift og styring av

heterogene kommunikasjonsnett. Vidare gir dei ein oversikt over det som faktisk er standard, og dermed det minste felles multiplum for ei stor mengd leverandørspesifikke implementasjonar og produkt som gjerne berre støttar deler av det totale området. Ulempa ved denne tilnærminga er at slike organisjonar og fora ikkje alltid er fremst i utviklingsarbeidet.

I dag er det i stor grad produsentane av kommunikasjonsutstyr som utviklar nye standardar ved å tvinga dei fram gjennom nye produkt. Ein kunne difor valt å gå inn i produktspesifikasjonar og konkrete system frå dei sentrale produsentane. Ein slik angrepsmåte ville nok kunna gitt betre kunnskap om kvar einskild implementasjon. Derimot ville framstillinga av kva drift og styring av kommunikasjonsnett allment handlar om truleg vorte dårlegare og meir fragmentert. Dessutan vil slike spesifikasjonar sjølvsagt vera nært knytt til resterande gammal og ny produktporteføljge hjå kvar einskild leverandør. Det ville difor vera vanskeleg å skildra drift- og styringsfunksjonaliteten isolert og uavhengig. Det ville truleg heller ikkje vera lett å få tilgang til relevant teknisk dokumentasjon.

1.2 Avgrensingar

System for handtering av kommunikasjonsnett kan kategoriserast etter graden av integrasjon med sjølve kommunikasjonsnettet (trafikknettet):

- *System som er integrert med trafikknettet*. Desse systema kommuniserer direkte med nettelementa i trafikknettet. Informasjon vert utveksla via standardiserte protokollar. Typisk informasjonsutveksling er alarmar og statistikkdata frå nettelementet eller konfigurasjons-data til nettelementet. Denne kommunikasjonen går som regel føre seg over separate samband i trafikknettet, og ein får dermed eit eige *tenestenett* overlagra trafikknettet. Nokre system verkar mot einskildelementa i nettet, medan andre òg kan handtera heile subnett, til dømes eit tenestenett eller eit geografisk avgrensa lokalnett. I praksis er desse systema framleis teknologispesifikke; dei er utvikla for særskilde transport- eller svitsjeteknologiar. Det er system i denne kategorien som hittil har vore omfatta av internasjonale standardar, og i denne rapporten er desse systema som vert kalla *drift- og styringssystem*.
- *System som er delvis integrert med trafikknettet*. Dette er system som til dømes berre mottek enkelte data frå trafikknettet. Det kan til dømes vera faktureringssystem som mottek tellardata fra telefon- eller datasentralar.
- *System som er skilt frå trafikknettet*. Dei mest sentrale nettadministrative systema kan innehalda ”totalinformasjon” om nettet:
  - Topologi og struktur ”på tvers av” transport- og svitsjeteknologiar.
  - Lokalisering av utstyr og kva for kapasitetar og eigenskapar det har.
  - Fysiske og logiske koplingar.
  - ”A-til-Å”-ruting av alle samband i og gjennom transportnettet.

  Slike system vert i første rekkje nytta for nettplanleggjing og ordrehandtering på alle nivå, men er òg viktige støttesystem i samband med den manuelle feillokaliseringsfunksjonen. I denne kategorien kjem òg geografiske informasjonssystem.

Det er glidande overgangar mellom *drift- og styringssystema* i den første og *støttesystema* i dei to andre kategoriane. Dei ulike aktørene kan dessutan ha ulike systemløysingar. Store aktørar kan ha stor grad av eigenutvikla system i dei to siste kategoriane. Graden av innbyrdes integrasjon mellom informasjonssystema samt av integrasjon mellom desse systema og sjølve trafikknettet, kan variera. I denne rapporten ser ein på arkitekturar for system som har

interaksjon med trafikknettet.

Dei digitale telefonsentralane sluttførte *digitaliseringa* av telenetta. Dette medførte ei sterk automatisering og sentralisering av driftsoppgåvene i dei tradisjonelle teleorganisasjonane. Eit heildigitalt telenett, drift- og styringssystem for dei sentrale transport- og svitsjeteknologiane samt "altomfattande" sentraliserte databasar i nettadministrative støttesystem, legg grunnlaget for at eit offentleg nasjonalt telenett i stor detalj kan planleggjast, styrast og drivast frå *ein* stad. I 1999 hadde Telenor att tre regionale driftssenter. Det er ikkje uvanleg at store operatørar set drifta av einskilde tenester og tenestenett ut til utanlandske samarbeidande operatørar. I denne rapporten ser ein ikkje på korleis dei ulike aktørane organiserer den operative verksemda. Ein ser heller ikkje på dei manuelle prosessane innan drift og styring av kommunikasjonsnett.

Å automatisera drift- og styringsfunksjonane for eit kommunikasjonsnett gjer i seg sjølv nettet meir sårbart. At dei manuelle drift- og styringsfunksjonane som framleis står att, i tillegg er sterkt sentraliserte, gjer det sjølvsagt endå lettare å skada nettet. I denne rapporten er tenester og mekanismar for datasikring i kommunikasjonsnett eit tema. Rapporten inneheld derimot ingen analyse av kor *sårbare* dei ulike drift- og styringskonsepta er. Dette krev ein omfattande analyse av mellom anna trugsmål, ulike organisasjonar og aktørar, ulike operative konsept så vel som av fysisk sikring av installasjonar og nett. Dette ligg utanfor denne studien.

1.3 Definisjonar

ISO, *International Electrotechnical Commissison* (IEC) og ITU-T har i samarbeid spesifisert OSI Environment (OSIE). Dei definerer *OSI Management* slik:

> *The facilities to control, coordinate and monitor the resources which allow communication to take place in the OSI environment* (9).

I denne rapporten vert denne definisjonen av *management* nytta, men ein avgrensar ikkje definisjonen til å gjelda berre OSI-miljø. Omgrepet *management* vert vidare omsett til *drift og styring*. Når ikkje anna er uttrykt, dreier deg seg om drift og styring av *kommunikasjonsnett*.

Ein oversikt over andre definisjonar og norske omsettingar som er nytta i rapporten finns i vedlegg A.

1.4 Oppbyggjing av rapporten

Kapittel 2 innheld ein gjennomgang av spesifikasjonane for OSI drift og styring.

Kapittel 3 gir ein oversikt over TMN som er basert på OSI-spesifikasjonane, men som òg representerer ei sterk utviding av desse. TMN er utvikla av ITU-T for å handtera store heterogene nett, tenester og utstyr. TMN skal operera på utstyr og teknologiar frå mange ulike leverandørar. TMN-baserte system dominerer i dag drift og styring av telenett.

Kapittel 4 inneheld ein oversikt over spesifikasjonane rundt SNMP. SNMP er utvikla av IETF for å handtera IP-nett. SNMP-baserte system dominerer i dag drift og styring av lokalnett så vel som Internett.

Dokumentasjonen som ligg til grunn for kapittel 2, 3 og 4 er svært omfangsrik og detaljert. I

rapporten prøver ein å skildra desse spesifikasjonane på ein kortfatta, men likevel heilskapleg og objektiv måte. Framstillinga kan nok difor verka noko tung og komprimert. Dette prøver ein å bøta på ved utstrekt bruk av figurar.

Kapittel 5 presenterer nyare arkitekturar for drift og styring av kommunikasjonsnett. Dette er arkitekturar som i større grad tek omsyn til konvergensen mellom tradisjonelle telenett og tradisjonelle datanett.

I kapittel 6 prøver ein å trekkja ut trendar i arbeidet med framtidige konsept for drift og styring av kommunikasjonsnett.

Kapittel 7 er ei kort oppsummering av rapporten.

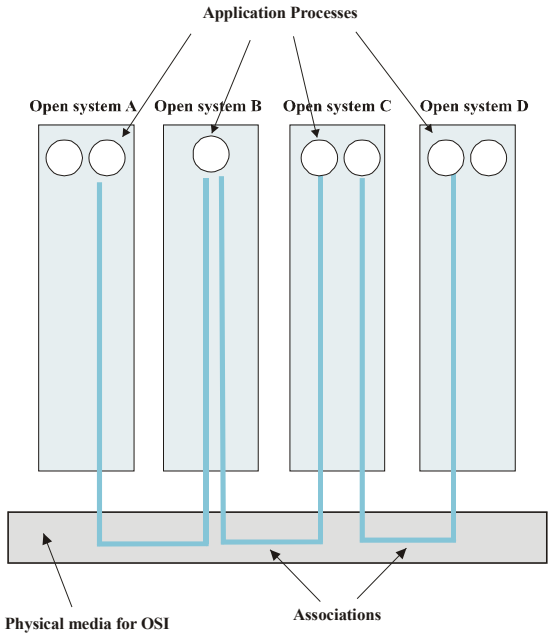
2 OPEN SYSTEM INTERCONNECTION (OSI) MANAGEMENT

Denne standarden er basis for implementasjonar først og fremst innanfor telekommunikasjon. Arkitekturen er basert på OSIE, og er langt meir kompleks enn tilsvarande arkitekturar som er utvikla for TCP/IP-nett.

Standarden er utvikla i samarbeid mellom ISO, IEC og ITU-T. Han er spesifisert i dels identiske og dels ekvivalente dokument i to ulike seriar: ISO/IEC International Standards og ITU-T Recommendations. Det er ITU-T-rekommendasjonane, serien X.700-X.799, som er nytta som kjelder i denne rapporten.

2.1 Generelt om Open System Interconnection (OSI) referansemodell

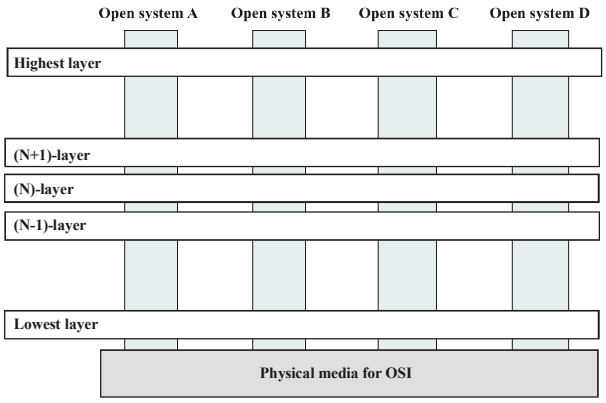
OSI referansemodell er skildra i (5). Siktemålet med standardar bygt på denne modellen, er å gjera kommunikasjon mellom autonome system mogleg. Eitkvart *verkeleg* system som kommuniserer i tråd med ein OSI protokoll-standard vil i modellkonseptet vera ekvivalent med eit såkalla *ope system*, sjå figur 2.1.



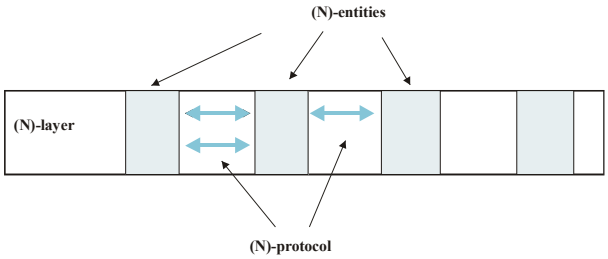
Figur 2.1 Opne system knytt saman ved fysisk medium (5)

OSI referansemodell er lagdelt. Det einskilde opne systemet er hierarkisk delt opp i (N)-subsystem. Eit slikt substystem kommuniserer direkte med subsystemet rett over og rett under i hierarkiet. Ser ein fleire ulike opne system under eitt, vil subsystem av same rang (N) utgjera (N)-laget i referansemodellen. (N)-laga, går ”på tvers” av dei ulike opne systema. Dette prinsippet er synt i figur 2.2.

Eit (N)-subsystem inneheld ein eller fleire (N)-entitetar som er element med definerte evner. Entitetar finns difor på kvart lag. Entitetar innan same (N)-lag kallast lag-(N)-entitetar. Dersom informasjon skal utvekslast mellom to eller fleire lag-(N)-entitetar, vert det oppretta ein assosiasjon mellom dei ved hjelp av ein protokoll. Ein (N)-entitet kan støtta fleire protokollar. (N+1)-entitetar kan kommunisera berre ved å nytta tenester frå lag (N).

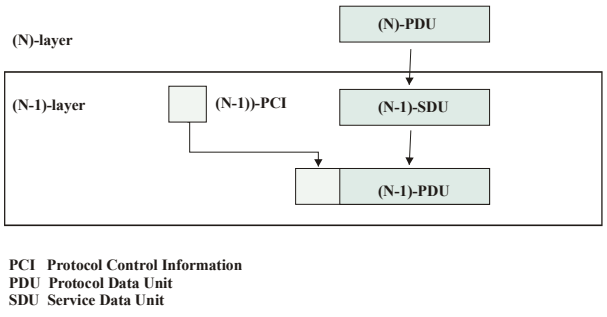


Figur 2.2 Lagdelinga i OSI referansemodell (5)



Figur 2.3 (N)-laget med (N)-protokollar mellom (N)-entitetar i ulike opne system (5)

Lag-(N)-entitetane overfører informasjonen i form av ulike typar dataeiningar. Samanhengen mellom desse einingane er illustrert i figur 2.4.



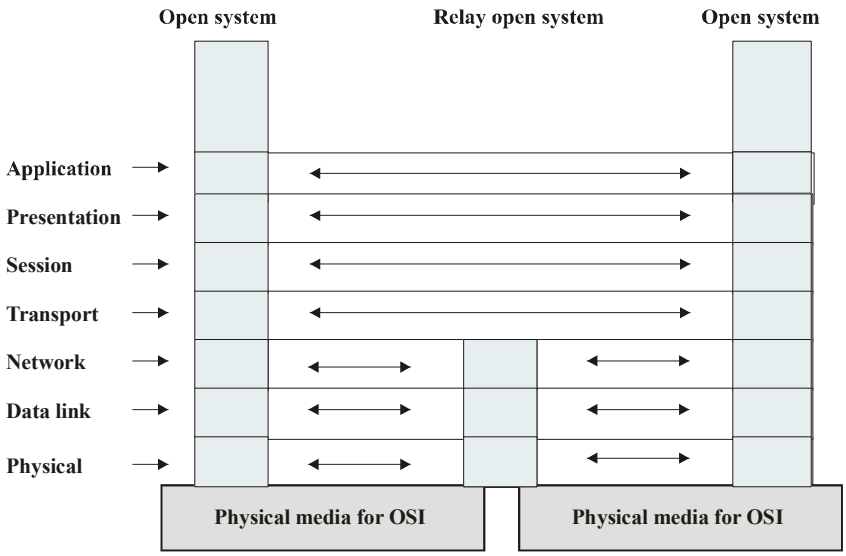
Figur 2.4 Dataeiningar i påfølgjande kommunikasjonslag (5)

Lag-(N)-entitetane kan kommunisera på to måtar:

- *Med tilknyting* (*Connection Mode*). Det vert sett opp ei ny (N)-tilknyting (*(N)-connection*) som krev eksplisitt identifikasjon av datatransmisjonen og semje vedrørande transmisjonstenestene som skal nyttast. Tilknytinga bind lag-(N)-entitetane til respektive (N-1)-entitetar. Tilknytinga har typisk tre fasar: oppkopling, overføring og nedkopling.
- *Utan tilknyting* (*Connectionless Mode*). Overføringa er basert på førehandsinformasjon om datatransmisjonen, og det vert ikkje sett opp noko tilknyting mellom lag-(N)-entitetane før

data vert overført. Dette dreier seg om enkeltvis overføring av data-einingar.

Det er definert sju lag i modellen: Fysisk lag, datalinklag, nettverkslag, transportlag, sesjonslag, presentasjonslag og applikasjonslag. Det høgaste laget, applikasjonslaget, leverer "alt som trengs" for at ein applikasjonsprosess skal få tilgang til OSIE. Laget er sett saman av applikasjonsentitetar som samhandlar innanfor OSIE. Ein applikasjonsentitet representerer ein, og berre ein, applikasjonsprosess. Ein slik prosess kan derimot representerast av fleire ulike applikasjonsentitetar. Desse entitetane utvekslar informasjon ved hjelp av applikasjonsprotokollar og tenester frå presentasjonslaget under. Dei andre laga leverer dei tenestene applikasjonsentitetane treng for å samhandla. Desse kommunikasjonstenestene vert utvida steg for steg for kvart lag oppover i modellen. Grensa mellom to lag i modellen identifiserer eit steg, og for kvart slikt steg er det definert ein OSI tenestestandard.



Figur 2.5 *OSI-kommunikasjon mellom opne system. I dette tilfellet er det òg med eit ope vidaresendingssystem (relay system) (5)*

2.2 Modell for OSI drift og styring

Drift og styring er essensielt for å få ulike system til å operera saman. *OSI Management Environment* er eit subsett av OSIE, og omfattar verktøy og tenester som trengs for å overvaka, styra og koordinera aktivitetar knytt til den innbyrdes samankoplinga av systema. Innan OSIE skal ein ha evne til å samla inn informasjon om ressursarne og å kunna styra og kontrollera dei. Likeeins skal ein vera i stand til å ha kjennskap til statusen til ressursane og å kunna rapportera denne. Ressursane er representert i modellen som *styrte objekt*.

Drift- og styring vert utført ved hjelp av eit sett av prosessar som manipulerer dei styrte objekta. Desse prosessane kan vera distribuert på ulike vis over ei rekkje system. Prosessane kommuniserer ved hjelp av drift- og styringsprotokollar. Berre prosessar som medfører *reell informasjonsutveksling* mellom opne system er relevante i standardiseringssamanheng, og det er difor berre dei protokollane som trengs for denne informasjonsutvekslinga som er standardiserte innanfor OSIE. Det er definert tre former for utveksling av drift- og styringsinformasjon, og det

finns standarder for alle tre:

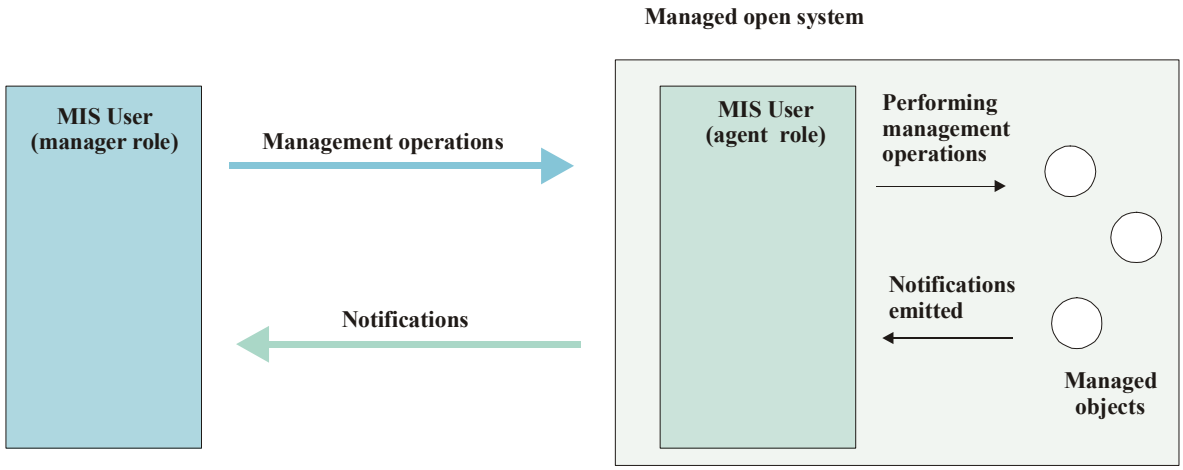
- *Drift og styring av system*. Dette er den vanlege forma for informasjonsutveksling og omfattar mekanismar for utveksling av informasjon relatert til monitorering, styring og koordinering av ressursar. Drift- og styringsprosessane nyttar i dette tilfellet drift- og styringsprotokollar frå applikasjonslaget.
- *Drift og styring av (N)-lag*. Dette vert nytta under særlege tilhøve for å overføra informasjon relatert til aktivitetar innan eit (N)-lag. Oftest vil dette forekoma på lag 2, 3 og 4, og ein nyttar gjerne spesialprotokollar for vedkomande lag for utveksling av informasjon.
- *(N)-lag operasjonar*. Utveksla informasjon er her avgrensa til *eitt* spesifikt kommunikasjonstilfelle innan eit (N)-lag.

Dei to sistnemnte formene vil ikkje verta omtalt vidare i denne rapporten.

Modellen for drift og styring er sett saman av fire submodellar som skildrar funksjonar, informasjon, kommunikasjon og organisasjon. Modellen er skildra i (9) og (10).

2.2.1 Manager og agent

Drift- og styringsapplikasjonar vert kalla *Management Information Service* (MIS*)-Users*. Interaksjonen i distribuerte applikasjonar går såleis føre seg mellom to *MIS-Users*, der den eine tek managerrollen og den andre agentrollen. Dette er synt i figur 2.6.



Figur 2.6 Interaksjon mellom manager og agent (10)

Agent er den delen av den distribuerte applikasjonen som har ein agentrolle. Agentrollen styrer objekt innan det *lokale* systemmiljøet. Ein agent utfører drift- og styringsoperasjonane på ”sine” objekt som ei følgje av at manager har kommunisert desse operasjonane. Ein agent kan òg motta notifikasjonar frå styrte objekt og vidaresenda desse til ein manager. Ein agent kan utveksla informasjon med fleire managers. Då vil agenten ha fleire agentrollar; ein for kvar assosiert managerrolle.

Manager er den applikasjonsdelen som har ein managerrolle. Managerrollen er ansvarleg for ein eller flere drift- og styringsaktivitetar. Manager spør agenten om å utføra drift- og styringsoperasjonar. Manager vil òg motta notifikasjonar frå agenten. Ein manager kan utveksla informasjon med fleire agentar. Då vil manager ha fleire managerrollar; ein for kvar assosiert agentrolle.

2.2.2 Funksjonsmodell

Det vert definert fem funksjonelle område for drift og styring (9):
- *Feilhandtering* som omfattar deteksjon, isolasjon og korreksjon av unormale operasjonar.
- *Bruksregistrering* som identifiserer kostnader og takserer bruken av ressursar i OSIE.
- *Konfigurering* der ein identifiserer, kontrollerer, samlar data frå og leverer data til opne system for å førebu, initialisera, starta og terminera tenester knytt til den innbyrdes samankoplinga av systema.
- *Yting* som omfattar evaluering av ressursane så vel som av effektiviteten i kommunikasjonen.
- *Datasikring* som omfattar allmenn støtte til gjeldande reglar for sikring av data og informasjon.

Kvart område inneheld ei rekkje funksjonar. Ein og same funksjon kan sjølvsagt inngå i fleire funksjonsområde. Ein agent kan vanlegvis ikkje avgjera kva som er formålet med ein operasjon som skal utførast eller med ein notifikasjon som skal returnerast. Agenten responderer på forespørslar frå manager utan å måtta kjenna konteksten.

Tjuefire funksjonar er spesifisert i kvar sin ITU-T-rekommendasjon. (1) gir ein god oversikt over funksjonsmodellen.

2.2.3 Informasjonsmodell

Informasjonsmodellen er fullstendig objektorientert og svært kompleks samanlikna med modellen som er utvikla for drift og styring av TCP/IP-nett. Strukturen på management informasjonen, *Structure of Management Information* (SMI), er spesifisert i tre rekommendasjonar: (14) skildrar konseptet for modellen, (15) spesifiserer dei ulike elementa i modellen og (16) gir retningslinjer for implementasjon.

Informasjonsbasen, *Mangement Information Base* (MIB), er sett saman av dei *styrte objekta* med sine attributtar. Eit styrt objekt er ein abstraksjon av ein ressurs i nettet. Ressursen kan vera ein lag-entitet, eit samband/kopling eller ei fysisk kommunikasjonseining. Eit styrt objekt kan vera spesifikt for *eitt* kommunikasjonslag, men kan òg vere relevant for fleire. Det kan vidare vera spesifikt for *ein* drift- og styringsfunksjon, eller det kan vera relevant for systemet som heilskap.

Kvart objekt er spesifisert med attributtar som skildrar eigenskapane til objektet. Attributtane har ein assosiert verdi som kan ha ein enkel eller kompleks struktur. Eit objekt vert vidare spesifiseret med eit sett av drift- og styringsoperasjonar som kan utførast på objektet, samt kva effekt kvar av desse operasjonen vil ha på objektet og attributtane. Dersom operasjonen har effekt på andre relaterte objekt, vert dette spesifisert. Operasjonane kan vera avhengige av tilstanden til objektet eller attributtane. Til slutt kan objekta senda ut notifikasjonar som inneheld informasjon vedrørande ei hending assosiert med objektet.

Mekanismane for å *kommunisera* operasjonar og notifikasjonar er standardiserte, i motsetning til mekanismane for å *utføra* operasjonar og notifikasjonar.

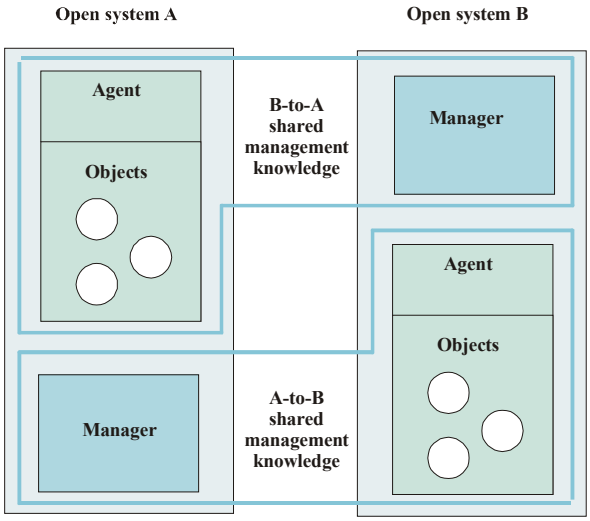
2.2.4 Kommunikasjonsmodell

For å utføra drift og styring av system, må manager og agent ha noko felles kunnskap. Dette kan til dømes dreia seg om kunnskap vedrørande protokollar, funksjonar, styrte objekt eller definisjonar.

Den felles kunnskapen kan etablerast til ei kvar tid:
- *Før kommunikasjon vert oppretta*. Naudsynt kunnskap for i det heile å mogleggjera kommunikasjon, er eit døme på slik førehandskunnskap.
- *I løpet av oppkoplingsfasen til assosiasjonen*. To applikasjonsentitetar etablerer ein assosiasjon ved å forhandla om ein applikasjonskontekst. Dette inneber mellom anna å identifisera den initialt delte kunnskapen som trengs for vedkomande assosiasjon.
- *I løpet av overføringsfasen til assosiasjonen*.

Interaksjonen mellom *MIS-Users* i rollane som manager og agent går føre seg gjennom informasjonsutveksling ved hjelp av OSI-protokollar.

Den allmenne OSI kommunikasjonstenesta for drift og styring er *Common Management Information Service* (CMIS). CMIS er spesifisert i (12). CMIS støttar kommunikasjon av operasjonar og notifikasjonar. Dette inneber:
- *Støtte til å overføra forespørslar om operasjonar og notifikasjonar mellom MIS-Users*.
- *Støtte til å utøva tilgangskontroll til styrte objekt*. Mekanismane for tilgangskontroll kan til dømes hindra spesifiserte managers i å utføra operasjonar på utvalde objekt.
- *Støtte til å spreia notifikasjonsinformasjon eksternt*. Det er definert ein mekanisme for å kunna identifisera destinasjonane for ekstern kommunikasjon.



Figur 2.7 Drift- og styringskunnskap som to system deler (10)

Protokollen CMIP vert nytta til dette. CMIP er spesifisert i (13).

2.2.5 Organisasjonsmodell

På dette området skal arkitekturen tilfredsstilla krav vedrørande handtering av:
- *Drift- og styringsreglar* (*management policy*). Dette vil først og fremst omfatta reglar for kva som *skal* gjerast (obligation) og kva som *kan* gjerast (authorization) med styrte objekt. Eit regelsett kan ikkje gå ut over det som er spesifisert i objektdefinisjonen. Eit regelsett er dynamisk, dvs det skal kunna etablerast, endrast og gå ut.
- *Drift- og styringsdomene* (*management domains*). Eit slikt domene er ei spesifisering eller samling av styrte objekt som sidan vert referert til som *medlemmar* i domenet. Eit objekt kan vera medlem i fleire domene, og medlemskapet er dynamisk. Objektet treng ikkje ”vita om” kvar det er medlem.

- *Drift- og styringsjurisdiksjon* (*management jurisdiction*). Dette kan sjåast som relasjonen mellom eit drift- og styringsdomene og eit sett drift- og styringsreglar. Eit regelsett vert knytt til styrte objekt ved at denne relasjonen vert oppretta. Relasjonen er dynamisk. Sidan eit objekt kan vera medlem i ulike domene, vil det såleis kunna verta "utsett for" ulike regelsett som kan vera i strid med kvarandre. Objektet treng ikkje "vita om" kva for regelsett det er knytt til.

2.3 Funksjonar

Ein funksjon har gjerne opphav i eitt av dei fem funksjonsområda, men kan sjølvsagt brukast innanfor dei andre. Nokre av funksjonane er svært komplekse. Det er utarbeidd standardar for ei rekkje funksjonar. Dei er vanlegvis definert generisk. Funksjonsspesifikke modellar er nytta for å definera funksjonane og kva for informasjon (objektklassar) som skal assosierast med funksjonen. Alle funksjonane kan avbildast til CMIS, sjå avsnitt 2.2.4. Tenestene i CMIS vert gjennomgått i eit seinare avsnitt.

2.3.1 Feilhandtering

Formålet med desse funksjonane er å kunna vedlikehalda og undersøkja feilloggar, godta og reagera på notifikasjonar vedrørande feildeteksjon, spora og identifisera feil, utføra sekvensar av diagnostiske testar og korrigera feil. Følgjande funksjonar er definert:

- *Alarm Reporting Function* definerer kva for notifikasjonar som skal sendast ved feil eller ulovlege hendingar. Definerte parametrar skal indikera moglege årsaker og grad av alvor. Det finns òg parametrar for terskelverdiar. Desse vert nytta ved alarmar som vert generert når terskelverdiar vert overskridne. Det er fem typar alarmrapportar: kommunikasjon, tenestekvalitet, prosessering, utstyr og miljø. Det er definert ei rekkje moglege årsaker til alarm.
- *Event Report Management Function* skal gjera det mogleg å spesifisera kva for vilkår som må tilfredsstillast for at ei hending skal rapporterast vidare til spesifiserte destinasjonar. Styrte objekt sender først notifikasjonar til ein lokal ikkje-standardisert mekanisme som prosesserer hendingar. Her samlast notifikasjonane opp til ein *potensiell* rapport. Denne blir så handsama av ein filtermekanisme, *Event Forwarding Discriminator* (EFD). EFD avgjer om den potensielle rapporten skal sendast, og i tilfelle kvar. Funksjonen definerer handteringa av EFD som dermed vert kontrollert av ein OSI drift- og styringsfunksjon.
- *Log Control Function* definerer operasjonar som samlar inn og lagrar notifikasjonar om og frå dei styrte objekta. Ein generisk modell er definert for loggar.
- *Test Management Function* definerer eit testmiljø med operasjonar for å starta og stoppa testar, samt format for utveksling av testresultat.
- *Confidence and Diagnostic Test Categories* definerer konkrete testkategoriar som skal støtta testing av tilgang til ressursane og yte-evna deira. Døme på testkategoriar er: ressursinterne testar, konnektivitetstestar, dataintegritetstestar, loop-, ekko- og protokolltestar.
- *Trouble Management Function* spesifiserer ein modell for problemhandtering i system og kommunikasjonsnett. Modellen skildrar korleis ei feilmelding frå brukar (av ei teneste) skal handterast av tenestetilbydar.

2.3.2 Bruksregistrering

Formålet med desse funksjonane er å kunna informera brukarane om kostnadane ved bruk av

ressursar, setja grenser for bruken, assosiera tariffar med bruken av ressursar samt kunna setja saman kostnadene ved bruk av fleire ressurser. Følgjande funksjonar er definert:

- *Usage Meterering Function for Accounting Purposes* definerer ei uniform skildring av bruksregistreringsdata og spesifiserer dei funksjonelle krava som er naudsynte for å kunna utveksla slik data på ein effektiv måte.

2.3.3 Konfigurering

Formålet med desse funksjonane er å kunna setja parametrar som styrer rutineoperasjonar, namngi objekt, setja opp og ta ned objekt, samla informasjon om tilstanden i systema og å endra konfigureringa av desse. Følgjande funksjonar er definert:

- *Object Management Function* definerer ei rekkje notifikasjonar som opprettar og slettar styrte objekt, og som endrar attributtane deira.
- *State Management Function* set opp ein generell tilstandsmodell og definerer operasjonar for å styra og monitorera tilstandsendringane til dei styrte objekta.
- *Management Relationship Function* understøttar etablering av og manipulasjon med relasjonar mellom styrte objekt.
- *Management Knowledge Management Function* tillet eit system å spørja eit anna om kva for drift- og styringskapasitetar dette systemet støttar. Dette kan vera kapasitetar relatert til styrte objektklassar, objektrelasjonar, namneskjema, domene, reglar og brukarar.
- *Changeover Function* skal handtera redundansen i nettet og støttar kommunisering av relasjonar mellom, og rollar til, styrte objekt. Rollane kan til dømes vera *active/standby* eller *backup/backed up*.
- *Software Management Function* blir brukt til å modellera aktivering og deaktivering av software, så vel som til å handtera interaktive aspekt ved nedlasting av software.
- *Management Domains and Management Policy Management Function* skal støtta etablering av domene og tildeling av reglar som skal følgjast.
- *Command Sequencer* tillet å delegera drift-og styringsfunksjonar til agentar. Funksjonen speglar OSI si tilnærming til *Management by Delegation* (MbD), som vert omtalt i kapittel 6.
- *Configuration Audit Support Function* definerer ein funksjon som let eit styrande system revidera konfigurasjon i eit styrt system. Det styrte systemet genererer ei fil som inneheld MIB-informasjonen i det styrte systemet. Slik informasjon kan ein sjølvsagt oppnå ved andre funksjonar, men denne funksjonen er langt raskare.

2.3.4 Yting

Formålet med desse funksjonane er å kunna samla statistisk informasjon, vedlikehalda og undersøkja loggar for tilstandshistorie og å få oversikt over kva systema yter i normale og unormale situasjonar. Følgjande funksjonar er definert:

- *Metric Objects and Attributes* definerer ein generisk modell for monitorering av terskelverdiar og spesifiserer funksjonalitet for monitorering av dynamiske attributtar og for å trigga alarmar når terskelverdiar vert overskridne.
- *Summarization Function* tillet eit agentsystem å preprosessera og redusera datamengda før data vert videresendt til manager. Statistiske algoritmar, som utrekning av gjennomsnitt og standardavvik, vert nytta i funksjonen.
- *Response Time Monitoring Function* måler *Protocol Data Unit* (PDU)-forseinking på eit forhåndsdefinert punkt-til-punkt- eller multipunktsamband på basis av ulike

evalueringsprosedyrar.

- *Scheduling Function* støttar tidsstyring av drift-og styringsoperasjoner. Dette gjeld spesielt funksjonalitet for å starta og stoppa operasjonar dagleg, kvar veke eller månad.
- *Time Management Function* definerer ei generisk tidsteneste som gjer bruk av tidssynkroniseringsprotokollar og ulike mekanismar for drift og styring av tidssynkronisering i nettet.

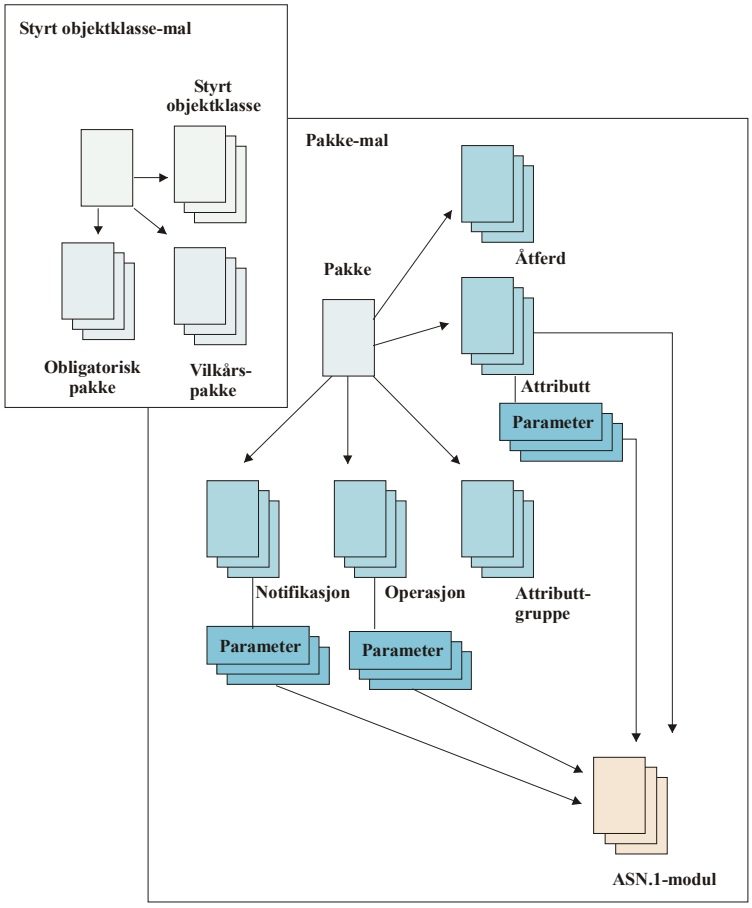
2.3.5 Datasikring

Formålet med desse funksjonane er å kunna oppretta, sletta og styra datasikringstenester og mekanismar, distribuera datasikringsinformasjon og å rapportera hendingar som er relevante for tryggleiken. Følgjande funksjonar er definert:

- *Objects and Attributes for Access Control* definerer operasjonar for å kunna introdusera og manipulera reglar for tilgangskontroll. Dette skal sikra at styrte objekt er verna mot uautoriserte, eksterne drift- og styringsoperasjonar. Funksjonen mogleggjer monitorering av uautoriserte, eksterne forsøk på informasjonsforespørslar. Det same gjeld for uautoriserte forsøk på å etablera kommunikasjon. *Access Control Decision Function* tillet bruk av ulike regelsett for datasikring. På basis av dette kan ein setja opp *Access Control Enforcement Function* mellom to partar for å sikra at eit regelsett vert følgt.
- *Security Alarm Reporting Function* er analog med *Alarm Reporting Function*, men tek seg særleg av saker som er relevante for drift og styring av datasikringa i nettet. Det er fem typar alarmrapportar: integritet, operasjon, fysisk øydeleggjing, sikringsmekanisme og tidsdomene. Dersom ein operasjon vert sett igang utanom tidsskjema, vil dette til dømes trigga ein tidsdomenealarm. Det er definert moglege årsaker til alarm. Endring av informasjon og forsøk på avlytting er nokre av årsakene.
- *Security Audit Trail Function* er ei videreutvikling av *Log Control Function*, men krava er her relatert til innsamling og lagring av sikringssrelevante notifikasjonar og operasjonar. Det vert generert revisjonar (*audits*) til spesielle loggar.

2.4 Informasjon

Figur 2.8 syner elementa i informasjonsmodellen.



Figur 2.8 Elementa i informasjonsmodellen (1)

2.4.1 Innkapsling, spesialisering og arv

For å sikra integriteten til objekta er det i objektorientert utforming vanleg å *kapsla dei inn*. Dette inneber at alle operasjonar som objektet skal utføra, kan setjast i gang berre ved å senda ei *melding* til objektet. Dei interne operasjonane i objektet er ikkje synlege utanfor objektgrensa, med mindre attributtar, operasjonar eller notifikasjonar er definert for å eksponera denne informasjonen. Definisjonen av ein styrt objektklasse spesifiserer kva for operasjonar som kan utførast på objekta og kva for konistenskrav som må tilfredsstillast for å oppretthalda integriteten til det styrte objektet.

Ein kan laga ein ny objektklasse, *y*, ved å utvida ein eksisterande objektklasse, *x*, med til dømes nye attributtar. *y* vert då ei *spesialisering* av *x*. Ein objektklasse som er spesialisert frå ein annan objektklasse kallast ein subklasse av den opprinnelege objektklassen, som no vert kalla superklasse. På dette viset kan ein byggja opp eit objektklassehierarki. Ein av fordelene med dette er at subklassane kan arva eigenskapane og karakteristikkane (attributtar, operasjonar, notifikasjonar og åtferd) frå superklassen. Informasjonsmodellen tillet vidare multippel arv som vil seia at ein subklasse kan arva eigenskapar frå fleire superklassar.

2.4.2 Objektklassar

Spesifikasjonen av ein objektklasse gir ein abstraksjon av verkelege prosesserings- og kommunikasjonsressursar i systemet sett frå ein drift-og styrings-vinkel. Ein definerer eigenskapar som er ”synlege” eksternt. Objektklassane er spesifisert på basis av *Abstract Syntax Notation One* (ASN.1). Definisjonen innheld følgjande element:

- Posisjonen objektklassen har i arvehierarkiet.
- Ei samling *obligatoriske* pakkar (*mandatory*) som kan innehalda attributtar, operasjonar, notifikasjonar og åtferd, sjå avsnitt 2.4.3.
- Ei samling *vilkårs*pakkar (*conditional*) som òg kan innehelda attributtar, operasjoner, notifikasjonar og åtferd, saman med vilkåra for at pakkane skal vera med.
- Innanfor pakkestrukturen:
  - dei eksternt synlege attributtane.
  - operasjonane som kan nyttast mot objektet eller attributtane.
  - notifikasjonane objektet kan senda ut.
  - åtferda objektet kan oppvisa.

Kvart objekt er ein instans av ein objektklasse. Ein objektklasse inkluderer alle objekt med dei same definisjonane. Definisjonane tillet allomorfi; eit objekt kan tilhøyra meir enn ein objektklasse. Dette gir ei rekkje fordeler ved praktisk drift- og styring av objektet.

Top er den ultimate superklassen. Klassen kan ikkje instansierast. Alle andre klassar er subklassar under denne. Objektklassane som er definert under *Top* er klassar som enten er referert til i funsksjonsspesifikasjonane og/eller det er klassar som er meint som superklassar for klassar i definert i andre spesifikasjonar. Følgjande generiske objektklassar vert definert:

- *System* er ein klasse under *Top*, og som representerer eit sett av hardware og software som formar ein autonom heilskap som er i stand til å prosessera og/eller overføra informasjon.
- *Log* er ein klasse under *Top*, og som definerer styringskriterium for logging av informasjon i eit ope system.
- *Log Record* er ein klasse under *Top*, som definerer postane i *Log*.
  - *Event Log Record* er ein subklasse under *Log Record* og definerer informasjonen som skal lagrast i loggen når ein mottek notifikasjon eller rapport vedrørande ei hending. Klassen er meint å vera superklasse for spesialisering av subklassar for ulike typar hendingar. Følgjande subklassar er definert under *Event Log Record*:
  - *Alarm Record* definerer informasjonen som skal lagrast i loggen når ein mottek alarmnotifikasjon eller alarmrapport.
  - *Attribute Value Change Record* definerer informasjonen som skal lagrast i loggen når ein mottek notifikasjon eller rapport vedrørande endring av attributtverdi.
  - *Object Creation Record* definerer informasjonen som skal lagrast i loggen når ein mottek notifikasjon eller rapport vedrørande oppretting av objekt.
  - *Object Delition Record* definerer informasjonen som skal lagrast i loggen når ein mottek notifikasjon eller rapport vedrørande sletting av objekt.
  - *Relationship Change Record* definerer informasjonen som skal lagrast i loggen når ein mottek notifikasjon eller rapport vedrørande endring av relasjonar.
  - *Security Alarm Report Record* definerer informasjonen som skal lagrast i loggen når ein mottek notifikasjon eller rapport om tryggingsalarm.
  - *State Change Record* definerer informasjonen som skal lagrast i loggen når ein mottek

notifikasjon eller rapport om tilstandsendring.
- *Discriminator* er ein klasse under *Top* og definerer kontrollkriterium for drift- og styringstenestene.
  - *Event Forwarding Discriminator* er ein subklasse under Discriminator og definerer vilkåra som skal tilfredsstillast for at ei hending skal medføra rapport til spesifisert destinasjon.

2.4.3 Pakkar

Ein pakke er ei samling karakteristikkar for ein objektklasse. Ein *obligatorisk* pakke er spesifikk for klassen, og må vera til stades for alle instansar av klassen. Ein *vilkårs*pakke er generell og kan vera til stades ved særlege vilkår. Desse vilkåra kan vera relatert til kapasitetane til den virkelige ressursen som objektet avbildar, eller ganske enkelt til om det aktuelle drift- og styringssystemet støttar funksjonaliteten. Dette medfører at objekt innan same objektklasse kan ha ulike sett attributtar avhengig av ulike vilkår.

Medan den obligatoriske pakken inneheld attributtar spesielle for objektklassen, er ein vilkårspakke allmenn og kan gå inn i definisjonen til ulike objektklassar. Ein attributt, notifikasjon eller operasjon kan likeeins gå inn i ulike pakkedefinisjonar. Følgjande vilkårspakkar vert definert:
- *Additional Information*
- *Additional Text*
- *Attribute Identifier List*
- *Attribute List*
- *Availability Status*
- *Correlated Notifications*
- *Notification Identifier*
- *Daily Scheduling*
- *Weekly Scheduling*
- *Duration*
- *External Scheduler*
- *Source Indicator*

2.4.4 Attributtar

Attributtane har ein assosiert verdi som kan vera eit sett eller sekvensar av element. Verdien kan vera observerbar. Verdien kan avgjera eller reflektera åtferda til objektet. Attributtverdien kan observerast eller modifiserast ved å senda ein forespørsel til objektet om å lesa (*get*) eller skriva over (*replace*) verdien. Det er definert tilleggsoperasjonar for å manipulera attributtar med sett-verdiar, dvs verdiar som er eit sett av element av same datatype. Operasjonar vert utført på *objektet* som inneheld attributtane, ikkje direkte på sjølve attributtane.

Definisjonen av objektklassen omfattar reglar som skal sikra at verdiar for individuelle attributtar er konsistente. Objektet er difor i stand til å sikra intern konsistens. Eit objekt skal til dømes ikkje returnera verdi utanfor lovleg verdisett og skal vidare avvisa forespørslar om å modifisera attributtar til ein verdi utanfor dette verdisettet. En har to typar restriksjonar på attributtverdiane:
- *Påkravd verdisett* som spesifiserer verdiane som attributten *må* vera i stand til å ha. (Dette

settet kan vera tomt viss det ikke finns krav.) Verdisettet er eit subsett av:
- *Lovleg verdisett* som spesifiserar verdiane som atributten *kan* ha.
- Sidan attributtane er definert til å vera ein del av ein pakke, vil dette medføra at objekt av same klasse ikkje nødvendigvis har dei same attributtane, sjå avsnitt 2.4.3.

Ein attributt er ein instans av ein attributtype. Ein skil mellom generiske og spesifikke attrubuttypar.

2.4.4.1 Generiske attributtypar

Ein generisk attributtype kan nyttast fritt i definisjonen av styrte objektklassar. Definisjonen omfattar:
- *Strukturen til attributtverdien*, til dømes ein einskildverdi eller eit sett av verdiar. Dersom attributten har ein sett-verdi, vert datatypane for kvar verdi lista, til dømes integer, boolean osv.
- *Ibuande eigenskapar*, til dømes initialverdi, inkrementverdi osv.
- *Lovlege operasjonar*, til dømes *Get attribute value*.
- *Implisitte relasjonar*, til dømes relasjon mellom tellarattributt og terskelattributt.
- *Spesifikasjonseigenskap*, til dømes maksimumsverdi.
- *Syntaks* som er ein ASN.1-type som skildrar korleis attributtverdien skal overførast i protokollar. Syntaksen er såleis bygd inn i attributten og vert verande konstant ved all bruk.

Følgjande generiske attributtypar er definert:
- *Counter* som er ein abstraksjon av underliggjande telleprosessar. Ein skil mellom tellarar som opererer autonomt, og tellarar der ein kan setja og endra verdiar gjennom operasjonar. Ein tellar vil alltid vera assosiert med ei *hending* som kan vera definert.
- *Gauge* som er ein abstraksjon av verdien til ein dynamisk variabel, til dømes talet på oppkoplingar som vert handtert av ein protokoll, eller endringsraten i ein trafikkteljar.
- *Treshold* som er den generelle mekanismen for å generera notifikasjonar på grunnlag av endringar i numeriske attributtverdiar. Ein har to typar terskelattributtar; ein type for *Counter*-tellar og ein for *Gauge*-tellar.
- *Tide-mark* er ein mekanisme som lagrar maksimum- og minimumsverdiar for ein gauge-attributt over eit definert tidsrom.

2.4.4.2 Spesifikke attributtypar

Ein spesifikk attributtype er knytt direkte til ein spesifikk objektklasse og er registrert med eigen attributt-ID. Semantikken er ofte skildra i ein av funksjonsrekommendasjonane. Dei viktigaste kategoriane av spesifikke attributtypar er:
- *Attributttypar for namngiving*. Denne kategorien vert nytta for å namngi instansar av objektklassar, til dømes attributtypen *LogID* som gir namnet på ein instans av objektklassen *Log*.
- *Attributtypar definert på grunnlag av generiske typedefinisjonar* frå forrige avsnitt.
- *Blanda attributtypar*. Desse attributtypane kan til dømes vera hendingsrelaterte, statusrelaterte eller relasjonsrelaterte.

2.4.5 Attributtgrupper

Gruppering av attributtar gjer det mogleg å referera til ei samling av attributtar innan eit objekt.

Det finns to typar attributtgrupper:
- *Endeleg attributtgruppe*, der attributtsettet ikkje kan endrast frå det initielle oppsettet. Alle einskildattributtane i ei slik gruppe må vera definert i same pakke som attributtgruppa.
- *Utvidbar attributtgruppe*, der attributtar kan leggjast til gruppa.

Ei attributtgruppe har ingen eigne verdiar. Berre operasjonar som ikkje krev at attributtverdiar skal spesifiserast, kan utførast på ei attributtgruppe. Ein objektklasse kan ha fleire attributtgrupper. Ein individuell attributt kan inkluderast i fleire attributtgrupper.

2.4.6 Åtferd

Definisjon av åtferda til objektklassens kan omfatta:
- Semantikk for attributtar, operasjonar og notifikasjonar.
- Respons på drift- og styringsoperasjonar som skal utførast på objektet.
- Vilkår for at notifikasjon skal sendast.
- Kor vidt spesielle attributtverdiar er innbyrdes avhengige.
- Effektar av relasjonar mellom objekt.
- Reglar for konsistens i attributtverdiar.
- Vilkår som på førehand må vera til stades for at ein kan gå ut frå at operasjonar og notifikasjonar har gyldig meining.
- Vilkår som må vera til stades for å kunna identifisera resultatet av operasjonsprosessering eller av utsending av notifikasjon.
- Invariantar som skal gjelda for heile levetida til objektet, og som skildrar vilkåra for operasjonar på objektet.
- Synkroniseringseigenskapar for objektet.

2.4.7 Operasjonar

Ein operasjon kan berre utførast dersom drift- og styringssystemet har tilgangsrettar til objektet. Dette vil vera avhengig av gjeldande reglar for datasikring. Både operasjonen og assosiert informasjon er ein del av definisjonen av objektklassen. Nokre operasjonar må stadfestast (*be confirmed*) av objektet.

Ein skil mellom attributtorienterte operasjonar som vedrører attributtane til objektet, og operasjonar som omfattar objektet som heilskap.

2.4.7.1 Attributtorienterte operasjonar

Følgjande operasjonar er definert:
- *Get attribute value*. Operasjonen er retta mot attributtar som er definert som lesbare. Lista med ønska attributtverdiar vert lesen og verdiane returnert. Operasjonen skal stadfestast.
- *Replace attribute value.* Operasjonen er retta mot attributtar som er definert som overskrivbare. Attributtverdiane vert erstatta med nye verdiar. Ein kan velja om operasjonen skal stadfestast.
- *Replace-with-default value*. Operasjonen er retta mot attributtar som er definert med defaultverdi som kan erstatta noverdi. Attributtverdiane vert erstatta med defaultverdi. Ein kan velja om operasjonen skal stadfestast.
- *Add member*. Operasjonen er retta mot settverdi-attributtar som er definert til å tillata nye medlemmar i settet. Attributtverdiane vert erstatta med unionen av noverande og nye

medlemmar. Ein kan velja om operasjonen skal stadfestast.
- *Remove member.* Operasjonen er retta mot settverdi-attributtar som er definert til å tillata sletting av medlemmar i settet. Attributtverdiane vert erstatta med differansen mellom noverande medlemmar og dei som skal slettast. Ein kan velja om objektet skal stadfesta operasjonen.

2.4.7.2 Operasjonar som omfattar objektet som heilskap

Følgjande operasjonar er definert:
- *Create*. Operasjonen vert nytta for å oppretta eit nytt styrt objekt, dvs ein instans av ein objektklasse. Operasjonen opprettar og initialiserer det nye objektet. Operasjonen skal alltid stadfestast.
- *Delete*. Operasjonen vert nytta for å sletta eit styrt objekt. Operasjonen skal alltid stadfestast.
- *Action*. Operasjonen kan brukast på alle styrte objekt. Objektet blir bedt om å utføra ein spesifisert aksjon og å indikera resultatet av denne. Ein kan velja om operasjonen skal stadfestast.

2.4.7.3 Atomisk synkronisering

Dersom ein operasjon skal utførast mot mange objekt, kan det kan vera krav om atomisk synkronisering. Dette vil seia at dersom ikkje alle operasjonane er vellukka, skal ingen utførast. For å kunna støtta dette, må difor alle operasjonar ha definerte kriterium for kor vidt dei er vellukka.

2.4.8 Notifikasjonar

Dei styrte objekta kan vera definert til å senda ut notifikasjonar når spesielle interne eller eksterne hendingar oppstår. Notifikasjonane er spesifikke for det objektet som sender dei. Om notifikasjonane vert sendt eksternt i protokoll eller berre vert loggført, er avhengig av korleis systemet er konfigurert. Først og fremst er dette avhengig av om notifikasjonen tilfredsstiller kriterium som vert sett i EFD, sjå *Event Report Management Function* i avsnitt 2.3.1. Det er ikkje definert kor vidt ein notifikasjon skal stadfestast.

Definisjonen av ein notifikasjonstype innheld strukturen på dataene som skal overførast i protokollen, strukturen på dataene som er resultat av hendinga og som skal overførast i protokollen, åtferda til notifikasjonen samt notifikasjons-ID.

Notifikasjonstypane må sjåast i samanheng med funksjonane frå avsnitt 2.3. Følgjande notifikasjonstypar vert definert:
- *Attribute Value Change*
- *Relationship Change*
- *State Change*
- *Communication Alarm*
- *Environmental Alarm*
- *Equipment Alarm*
- *Processing Error Alarm*
- *Quality of Service Alarm*
- *Integrity Violation*
- *Operational Violation*

- *Physical Violation*
- *Security Service or Mechanism Violation*
- *Time Domain Violation*
- *Object Creation*
- *Object Deletion*

2.4.9 Parametrar

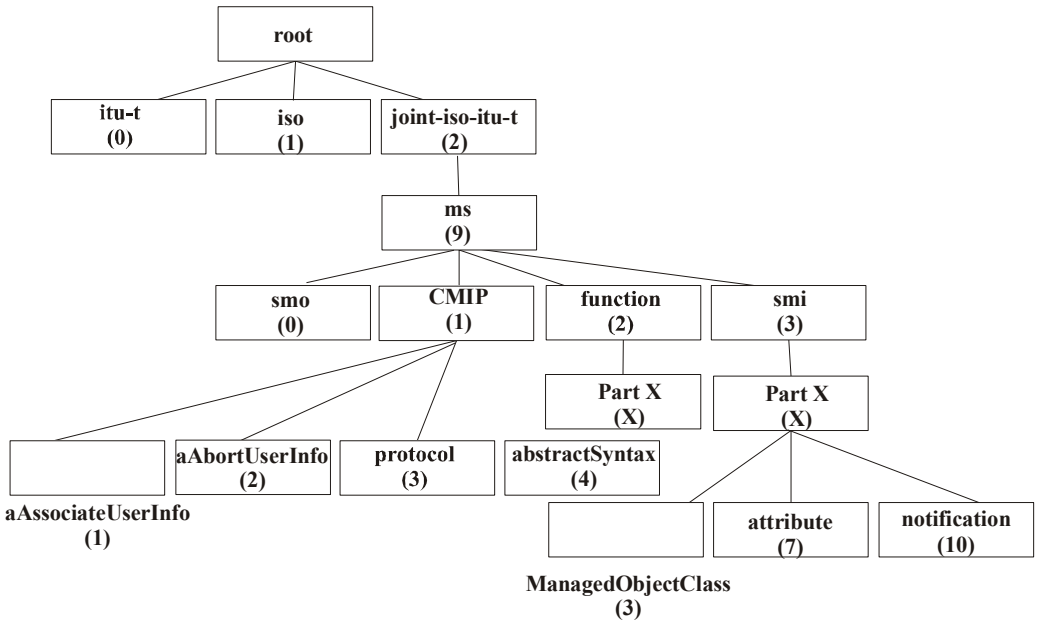
Parametrar er verdiar som kan overførast i ein protokoll. Parametrar er definert med semantikk og verditype. Parametrar kan assosierast med attributtar, operasjonar og notifikasjonar.

2.4.10 Strukturering

Informasjonen kan strukturerast i høve til tre ulike trestrukturar:

2.4.10.1 Registration Tree

Denne strukturen er ein katalogstruktur introdusert av ISO/ITU-T. Strukturen gjer det mogleg å gi eit kvart objekt frå eit kvart domene ei unik global adresse. Dei interne nodane i treet er nytta for strukturering; berre lauva i treet innheld konkret informasjon. Informasjonen kan vera abstraksjonar av nettverkselement (styrte objekt), så vel som standarddokument. Figur 2.9 syner deler av toppen av denne strukturen.



Figur 2.9 Registration tree

ISO/ITU-T er i fellesskap ansvarleg for nummerering av alle nodar under *joint-iso-itu*.
Ein finn den standardiserte drift- og styringsinformasjonen under *ms*-noden 2.9. Under *smo*-noden finns rekommendasjonar som syner systemoversikt. Under *cmip*-noden ligg spesifikasjonar vedrørande protokollen, under *function* funksjonsspesifikasjonar og til slutt spesifikasjonane for informasjonsstruktur under *smi*-noden. På det lågaste nivået her ligg identifikatorane for dei ulike objektklassane, pakkane osv.

2.4.10.2 Inheritance Tree

Denne strukturen syner korleis dei ulike objektklassane er relatert til kvarande gjennom arv av eigenskapar, sjå avsnitt 2.4.1. Treet syner klassehierarkiet under *Top* som er rota til dette treet, sjå avsnitt 2.4.2.

2.4.10.3 Management Information Tree (MIT)

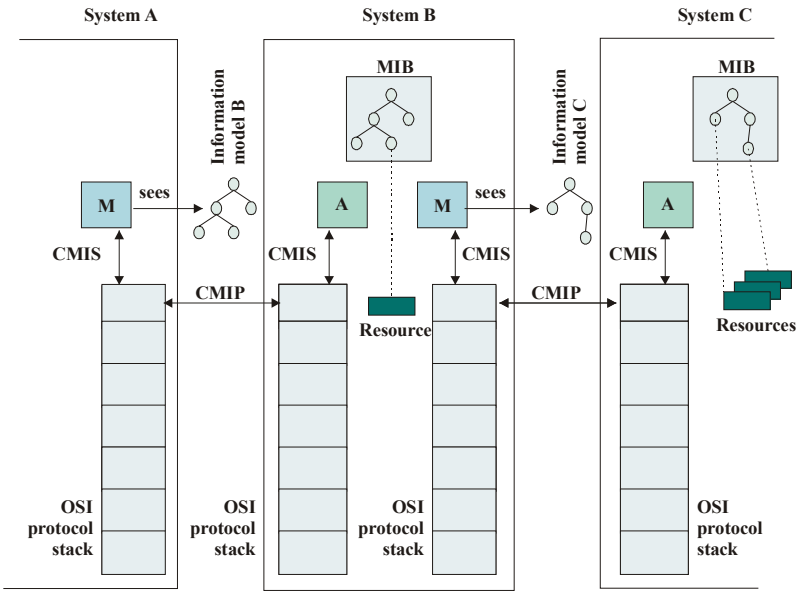
Dette treet avbildar sjølve MIB-strukturen i systemet. Eit styrt objekt av ein klasse kan innehalda objekt av ein annen klasse. Denne relasjonen kallast *containment* og er ein relasjon mellom einskildobjekt, ikkje mellom klassar. Ved hjelp av denne relasjonen kan ein modellera hierarki i det verkelege kommunikasjonsnettet/systemet.

Relasjonen vert òg nytta for å setja namn på objekta, og vert då kalla namnebinding (*name binding*). Objektnamna er unike innanfor ein kontekst. Det er det objektet som inneheld andre objekt som styrer namngivinga. Dette objektet er *superior object* til sine *subordinate objects*. Namnet til *subordinate object* vil då vera sett saman av namnet til *superior object* og informasjon som identifiserer objektet unikt innanfor konteksten til *superior object*.

En namnekontekst vert difor rekursivt kvalifisert av ein annen namnekontekst slik at den komplette namnestrukturen kan sjåast som eit hierarki med ei rot. Dette hierarkiet vert kalla *namnetre*. Namneattributtane kallast *Relative Distinguished Name* (RDN). Når ein konkatenerer RDNs nedover i treet, får ein *Distinguished Name* (DN), som identifiserer kvart einaste objekt unikt. Denne prosedyren er kjent frå katalogtenesta X.500.

2.5 Kommunikasjon

Figur 2.10 syner elementa i kommunikasjonsmodellen.



Figur 2.10 Interaksjon mellom managers og agentar (18)

Figuren syner korleis system A styrer system B som igjen styrer system C. Dette kallast kaskade av system.

- System A samhandlar med system B ved å referera til informasjonsmodellen som vert støtta av system B. Det same skjer mellom system B og system C.
- System B skal presentera sin informasjonsmodell til system A, men for å gjera dette må system B bruka informasjon frå system C.
- System B prosesserer operasjonar frå system A på MIB-objekt i B. Dette kan vidare medføra operasjonar mot MIB-objekt i C.
- System B prosesserer notifikasjonar frå system C. Dette kan vidare medføra notifikasjonar til system A.

2.5.1 Assosiasjonstenester

For å etablera, avslutta eller abortera CMIP-assosiasjonar mellom applikasjonar, nyttar ein tenester frå *Assosiacion Control Service Element* (ACSE) (6). For å etablera ein assosiasjon, brukar ein tenesta *A-Associate* med følgjande CMIS-spesifikke parametrar:

- *Funksjonelle einingar* som skal nyttast i assosiasjonen. Dette har med støtta til teneste-primitiv og parametrar å gjera.
- *Tilgangskontroll* for å verifisera rettane til den som initierer assosiasjonen.
- *Brukarinformasjon* som kan vera spesifikk for kver einskild applikasjon.

Tenestene *A-Release* og *A-Abort* vert nytta for å ta ned assosiasjonen.

2.5.2 Informasjonsoverføringstenester for operasjonar og notifikasjonar

Det er to typar informasjonsoverføringstenester: notifikasjonstenester (*M-Event-Report*) og operasjonstenester (dei resterande). Kvar teneste er definert med:

- *Parametrar* som spesifiserer informasjonen som skal overførast i protokollen
- *Prosedyrar* som spesifiserer korleis ein operasjon skal utførast på objektet/objekta.

CMIS Entity (CMISE)-tenestene er lista i tabell 2.1.

Service	**Type**	**Kommentar**
M-CANCEL-GET	Confirmed	Tenesta vert brukt for å kansellera ei *M-Get*-teneste.
M-EVENT-REPORT	Confirmed/ non-confirmed	Tenesta vert initiert av ein prosess når ei hending oppstår, sjå avsnitt 2.4.8.
M-GET	Confirmed	Tenesta vert brukt for å få informasjon om eit objekt, sjå operasjonen *Get attribute value* i avsnitt 2.4.7.
M-SET	Confirmed/ non-confirmed	Tenesta vert brukt for å endra informasjon om eit objekt, sjå operasjonane *Replace attribute value, Replace with default value, Add member* og *Remove member*.
M-ACTION	Confirmed/ non-confirmed	Tenesta vert brukt for å få eit objekt til å utføra ein aksjon, sjå operasjonen *Action*.
M-CREATE	Confirmed	Tenesta vert brukt for å oppretta eit objekt, sjå operasjonen *Create*.
M-DELETE	Confirmed	Tenesta vert brukt for å sletta eit objekt, sjå operasjonen *Delete*.

Tabell 2.1 CMISE-tenester (12)

Tenestene har tilgang til informasjonsbasen og namnetreet som vart skildra i avsnitt 2.4. Dei

kan relaterast direkte til operasjonane og notifikasjonane som vart gjennomgått i avsnitta 2.4.7 og 2.4.8. Ved ei operasjonsteneste vil mottakar validera parametrane. Om feil vert detektert, vert feilkodar returnerert. Ellers vert operasjonen utført, og resultatet vert returnert dersom operasjonen krev dette.

Ein har fire typar primitiv: *Response*, *Confirm*, *Request* og *Indication*. Desse vert assosiert med kvar av tenestene. Tabell 2.2 syner kva for parametrar som vert nytta for dei ulike tenesteprimitiva.

Teneste / Parameter	M-Event-Report		M-Get		M-Set		M-Action		M-Create		M-Delete		M-Cancel-Get	
	Req Ind	Resp Conf	Req Ind	Resp Conf	Req Ind	Resp Conf	Req Ind	Resp Conf	Req Ind	Resp Conf	Req Ind	Resp Conf	Req Ind	Resp Conf
Access control	-	-	U	-	U	-	U	-	U	-	U	-	-	-
Action information	-	-	-	-	-	-	U	-	-	-	-	-	-	-
Action reply	-	-	-	-	-	-	-	C	-	-	-	-	-	-
Action type	-	-	-	-	-	-	M	C	-	-	-	-	-	-
Attribute identifier list	-	-	U	-	-	-	-	-	-	-	-	-	-	-
Attrinute list	-	-	-	C	-	U	-	-	U	C	-	-	-	-
Base object class	-	-	M	-	M	-	M	-	-	-	M	-	-	-
Base object instance	-	-	M	-	M	-	M	-	-	-	M	-	-	-
Current time	-	U	-	U	-	U	-	U	-	U	-	U	-	-
Errors	-	C	-	C	-	C	-	C	-	C	-	C	-	C
Event information	U	-	-	-	-	-	-	-	-	-	-	-	-	-
Event reply	-	C	-	-	-	-	-	-	-	-	-	-	-	-
Event time	U	-	-	-	-	-	-	-	-	-	-	-	-	-
Event type	M	C	-	-	-	-	-	-	-	-	-	-	-	-
Filter	-	-	U	-	U	-	U	-	-	-	U	-	-	-
Get invoke identifier	-	-	-	-	-	-	-	-	-	-	-	-	M	-
Invoke identifier	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Linked identifier	-	-	-	C	-	C	-	C	-	-	-	C	-	-
Managed object class	M	U	-	C	-	C	-	C	M	C	-	C	-	-
Managed object instance	M	U	-	C	-	C	-	C	U	C	-	C	-	-
Mode	M	-	-	-	M	-	M	-	-	-	-	-	-	-
Modification list	-	-	-	-	M	-	-	-	-	-	-	-	-	-
Reference object inst.	-	-	-	-	-	-	-	-	U	-	-	-	-	-
Scope	-	-	U	-	U	-	U	-	-	-	U	-	-	-
Superior object instance	-	-	-	-	-	-	-	-	U	-	-	-	-	-
Synchronization	-	-	U	-	U	-	U	-	-	-	U	-	-	-

M	Mandatory
C	Conditional
U	User option
-	Not applicable
Req Ind	Request Indication
Rsp Conf	Response Confirm

Tabell 2.2 Parametrar som vert nytta i CMISE-tenester (1)(12)

To av desse parametrane gjer det mogleg å finna dei relevante objekta på ein effektiv måte:

- *Scope*. Parametrane Base Object Class/Instance er definerer klassen/instansen som skal vera rot i eit subtre i namnetreet, sjå avsnitt 2.4.10.3. Dette vert så starten på søket etter ønska objekt.

- *Filter*. Filter vert nytta for å spesifisera kriterium som eit styrt objekt må fylla for at ein operasjon skal utførast. Operasjonen vert berre utført om objektet fyller vilkåra. Filtra nyttar definerte testar som evaluerer om relevante attributtar er til stades og/eller verdiane deira. Parameteren *Filter* spesifiserer eit sett av slike testar som skal utførast på objekta i subtreet for å sila ut eit nytt subsett av godkjente objekt. Operasjonen blir så utført mot objekta i dette subsettet.

Parameteren *Synchronize* kan òg nemnast. Denne spesifiserer på kva måte operasjonane skal synkroniserast når ein operasjon skal utførast mot fleire objekt:
- *Atomisk synkronisering* medfører at operasjonane berre vert utført dersom det let seg gjera å utføra han på *alle* dei utvalde objekta.
- *Beste evne-synkronisering* medfører at operasjonane vert utført på objekt der det let seg gjera.

2.5.3 Protokoll

Dei to CMIS tenesteprimitiva, *request* eller *respons*, vert oversendt CMIP protokollmaskin som formar CMIP-PDUs. Likeeins vil protokollmaskina godta innkomande CMIP-PDUs og vidaresenda desse, til relevant CMISE-applikasjon for prosessering, ved hjelp av primitiva *indication* eller *confirmation*.

(13) spesifiserer prosedyrar for alle assosiasjons- og informasjonstenestene som vart skildra i forrige avsnitt. Prosedyrane har reglar for korleis sendar skal konstruera PDUs og korleis mottakar skal handtera og evt svara på dei. PDUs er spesifisert ved hjelp av ASN.1. For kvar CMISE-parameter vil det vera eit PDU-felt med same namn som korresponderande parameter.

CMIP nyttar tenester frå *Remote Operation Service Element* (ROSE) (7) for å overføra operasjonane som skal utførast på dei styrte objekta. CMIP kan baserast på ulike underliggende protokollstakkar, OSI-stakk så vel som TCP-IP.

Tabell 2.3 syner samanhengen mellom CMIS-primitiva og CMIP-operasjonane.

CMIS primitive	Mode	Linked-ID	CMIP operation
M-CANCEL-GET req/ind	Confirmed	Not applicable	m-Cancel-Get-Confirmed
M-CANCEL-GET rsp/conf	Not applicable	Not applicable	m-Cancel-Get-Confirmed
M-EVENT-REPORT req/ind	Non-confirmed	Not applicable	m-EventReport
M-EVENT-REPORT req/ind	Confirmed	Not applicable	m-EventReport-Confirmed
M-EVENT-REPORT rsp/conf	Not applicable	Not applicable	m-EventReport-Confirmed
M-GET req/ind	Confirmed	Not applicable	m-Get
M-GET rsp/conf	Not applicable	Absent	m-Get
M-GET rsp/conf	Not applicable	Present	m-Linked-Reply
M-SET req/ind	Non-confirmed	Not applicable	m-Set
M-SET req/ind	Confirmed	Not applicable	m-Set-Confirmed
M-SET rsp/conf	Not applicable	Absent	m-Set-Confirmed
M-SET rsp/conf	Not applicable	Present	m-Linked-Reply
M-ACTION req/ind	Non-confirmed	Not applicable	m-Action
M-ACTION req/ind	Confirmed	Not applicable	m-Action-confirmed
M-ACTION rsp/conf	Not applicable	Absent	m-Action-confirmed
M-ACTION rsp/conf	Not applicable	Present	m-Linked-Reply
M-CREATE req/ind	Confirmed	Not applicable	m-Create
M-CREATE rsp/conf	Not applicable	Not applicable	m-Create
M-DELETE req/ind	Confirmed	Not applicable	m-Delete
M-DELETE rsp/conf	Not applicable	Absent	m-Delete
M-DELETE rsp/conf	Not applicable	Present	m-Linked-Reply

Tabell 2.3 CMIS-primitiva og CMIP-operasjonane (13)

2.6 Datasikring

Datasikring i drift- og styring av kommunikasjonsnett kan drøftast frå to synsvinklar:

- Eit utgangspunkt er å sjå på korleis drift- og styringssystemet handterer sikringstenestene i det underliggjande kommunikasjonsnettet. I OSI-konseptet er det i stor grad ei drift- og styringsoppgåve å handtera sikringsreglane som skal gjelda innanfor eit drift- og styringsdomene. Dette dreier seg til dømes om korleis sikringstenester og mekanismar kan konfigurerast via eit drift- og styringssystem, og korleis åtak på kommunikasjonen/ kommunikasjonsnettet kan oppdagast og rapporterast. Datasikringsfunksjonane vart presentert i avsnitt 2.3.5. Desse vil sjølvsagt vera avhengige av kva tenester og mekanismar entitetane i sjølve kommunikasjonsnettet tilbyr. I dette avsnittet vil ein sjå på den generelle datasikringa som er spesifisert for OSI. Informasjonsflyten i drift- og styringssystemet vil vera underlagt og verna av denne.
- Eit anna utgangspunkt kunne vera å sjå på om, og i tilfelle korleis, informasjonsflyten mellom komponentar i drift- og styringssystemet er sikra *spesielt*. Denne informasjonen vil flyta i det allmenne kommunikasjonsnettet, men kan i utgangspunktet sikrast spesielt gjennom gjennom ”strengare” konfigurering av dei allmenne sikringstenestene/mekanismane i nettet, krav til ruting, mekanismar i protokollane og anna. Dette vil vera eit utgangspunkt når ein i neste kapittel ser på datasikringa i TMN.

(17) gir ein oversikt over OSI-arkitekturen for datasikring. Basistenester kan finnast på ulike kommunikasjonslag og i ulike kombinasjonar. Ein skil mellom *tenestene* på den eine sida og dei *mekanismane* som implementerer desse tenestene på den andre. Dei ulike trugsmåla som sikringstenestene skal handtera samt ulike aspekt rundt reglane vert òg skildra.

2.6.1 Sikringstenester

Det er definert tenester på følgjande område:

- *Autentisering*. Her er to tenester definert:
  - Autentisering av lag-entitet. Dette skal stadfesta at ein lag-entitet i ein assosiasjon er den han hevdar å vera. (N)-laget leverer tenesta til ein (N+1)-entitet for å stadfesta at (motståande) lag-entitet er den påståtte (N+1)-entiteten. Tenesta kan leverast både som einvegs- og tovegs-teneste.
  - Autentisering av datakjelde. Dette skal stadfesta at kjelda for mottekne data er den ho hevdar å vera. (N)-laget leverer tenesta til ein (N+1)-entitet for å stadfesta at kjelda er den påståtte lag-(N+1)-entiteten. Tenesta omfattar ikkje vern mot duplisering eller modifisering av dataeiningane.
- *Tilgangskontroll*. Tenesta skal sikra at uautoriserte ikkje får tilgang til ein ressurs. Tenesta omfattar òg vern mot at ressurene vert brukt på ein uautorisert måte. Tilgangskontrollen kan gjelda både OSI- og ikkje-OSI-ressursar som er tilgjengelege via ein OSI-protokoll. Vernet kan gjelda alle former for tilgang til ressursen eller berre til utvalde tilgangsmåtar, til dømes lesing, skriving, operasjonar, sletting.
- *Datakonfidensialitet*. Tenestene skal sikra at informasjon ikkje vert gjort tilgjengeleg eller avslørt for uautoriserte individ, entitetar eller prosessar. Ein har definert fire tenester:
  - Konfidensialitet i overføring med tilknyting (*Connection confidentiality*). Dette omfattar alle (N)-brukardata i ei (N)-tilknyting.
  - Konfidensialitet i overføring utan tilknyting (*Connectionless confidentiality*). Dette omfattar alle (N)-brukardata i ei einskild (N)-*Service DataUnit* (SDU).
  - Konfidensialitet for utvalde felt. Dette omfattar utvalde felt i (N)-brukardata i overføring med tilknyting så vel som i ein einskild (N)-SDU i overføring utan tilknyting.
  - Konfidensialitet på trafikkflyt. Dette skal verna informasjon som elles kunne vorte trekt ut ved observasjon av trafikkflyt.
- *Dataintegritet*. Tenestene skal sikra at data ikkje vert endra eller øydelagt på ein uautorisert måte. Tenestene skal møta aktive trugsmål. Ein har definert fem tenester:
  - Integritet med gjenvinning i overføring med tilknyting (*Connection integrity with recovery*). Dette omfattar alle (N)-brukardata i ei (N)-tilknyting. Tenesta detekterer eit kvart tilfelle av modifikasjon, innsettjing, sletting og duplisering av data innanfor ein heil SDU-sekvens. Tenesta prøver å gjenvinna data.
  - Integritet i overføring med tilknyting (*Connection integrity*). Som over, men gjenvinning vert ikkje forsøkt.
  - Integritet for utvalde felt i overføring med tilknyting. Dette omfattar utvalde felt i (N)-brukardata i ein (N)-SDU. Tenesta avgjer om data i dei utvalde felta har vorte modifisert, innsett, sletta eller duplisert.
  - Integritet i overføring utan tilknyting (*Connectionless integrity*). Dette omfattar ein einskild SDU og kan avgjera om motteken SDU er modifisert. Ei enkel form for deteksjon av duplisering kan utførast.
  - Integritet for utvalde felt i overføring utan tilknyting. Tenesta sørgjer for integritet for utvalde felt innan ein einskild SDU og kan avgjera om felta i motteken SDU er modifisert.
- *Ikkje-fornekting* (*Non-repudiation*). Tenestene skal sikra at entitetane som er involverte i kommunikasjonen ikkje kan nekta for å ha delteke i heile eller deler av kommunikasjonen. Det er definert to tenester:

- Ikkje-fornekting med prov på kjelde. Mottakeren får prov på at kjelda har sendt dataene. Dette vernar mot at sendaren kan nekta å ha sendt dataene.
- Ikkje-fornekting med prov på levering. Senderen får prov på at mottakar har motteke dataene. Dette vernar mot at mottakar kan nekta å ha motteke dataene.

2.6.2 Sikringsmekanismar

Følgjande sikringsmekanismar kan vera bygd inn i eit (N)-lag for at laget skal kunna levera sikringstenestene som vart gjennomgått i forrige avsnitt:

- *Kryptering* gir konfidensialitet for data så vel som for trafikkflyt, og kan spela ein viktig rolle i dei andre mekanismane som er skildra nedanfor. Krypteringsalgoritmane kan grovt delast i:
  - Symmetriske algoritmar med hemmeleg nøkkel.
  - Asymmetriske med offentleg nøkkel.
- *Digital signatur*-mekanismane definerer to prosedyrar:
  - Signera dataeining. Denne prosessen nyttar informasjon som er privat (unik og konfidensiell) for den som signerer. Informasjonen vert nytta til enten å kryptera dataeininga eller til å produsera ein kryptografisk sjekkverdi for dataeininga. Den private informasjonen vert som ein privat nøkkel.
  - Verifisera ei signert dataeining. Denne prosessen nyttar informasjon som er offentleg tilgjengeleg for å avgjera om signaturen verkeleg vart produsert med den private informasjonen til signeraren.

  Det essensielle ved digital signatur er at signaturen kan produserast berre med den private informasjonen til signeraren. Når signaturen er verifisert, kan det til ei kvar tid provast at berre den som har hatt den private informasjonen, kan ha produsert signaturen.
- *Mekanismar for tilgangskontroll* kan vera basert på:
  - Informasjonsbasar som inneheld tilgangsrettane for entitetane.
  - Autentiseringsinformasjon som til dømes passord.
  - Tidspunkt for forsøk på tilgang.
  - Kor lengje tilgangen varer.

  Mekanismen kan verta nytta i beggje endar og/eller på eit kvart mellomliggjande punkt.
- *Mekanismar for data-integritet*. Det vert nytta ulike mekanismar avhengig av om det dreier seg om integritet for eit einskild felt, ei einskild dataeining eller for ein heil datastraum. Dersom det gjeld ei einskild dataeining, vil det på sendarsida verta lagt til ein kvantitet som er ein funksjon av dataene sjølv. Dette kan til dømes vera ein sjekkverdi som evt kan krypterast. Mottakersida genererer ein korresponderende kvantitet, og ved å samanligne kvantitetane kan ein detektera om dataeininga er modifisert. (Denne mekanismen åleine vil korkje detektera eller verna mot at dataeininga er duplisert.) Dersom det dreier seg om integriteten til ein heil datastraum, må ein i tillegg ha mekanismar som ordnar sekvensane, tidsstempel osv.
- *Mekanismar for utveksling av autentiseringsinformasjon*. Her kan fleire teknikkar nyttast, til dømes kryptering. Mekanismane kan vera bygd inn i (N)-lag for å sørgja for autentisering av lag-entitetar. Val av mekanisme vil vera avhengig av om tidsstempel og synkroniseringsklokker er tilgjengelege, om utvekslinga er ein- eller tovegs, og korleis ei evt ikkje-fornektingsteneste er implementert.
- *Trafikkfylling* (*padding*) vert brukt for å hindra trafikkanalyse. Mekanismen er effektiv berre dersom han vert verna av ei konfidensialitetsteneste.

- *Mekanismar for styring av ruting*. Ruting (dynamisk eller statisk) kan til dømes setjast slik at ein berre nyttar subnett og linkar som er sikra fysisk.
- *Mekanismar for bruk av notar* (*notarization*). Mekanismen byggjer på at ein tredjeparts notar stadfestar eigenskapar til dataene som vert overført mellom to eller fleire entitetar. Desse eigenskapane kan til dømes vera integritet, tidspunkt, kjelde og destinasjon. Dei kommuniserende partane må stola på tredjepart (*trusted third party*). Notar har naudsynt informasjon til å kunna stadfesta informasjonen på ein verifiserbar måte. Kommunikasjonspartane kan bruka digital signatur, kryptering og integritetsmekanismar. Når notarmekanismen trår i kraft, går alle data mellom partane via notar.

I tillegg til desse mekanismane vert det spesifisert mekanismar for å indikera kor sensitiv ein ressurs er, for å detektera sikringsrelevante hendingar, for å revidera datasikringa og for å kunna gjenoppretta tryggleiken ved å setja i verk aksjonar som å avbryta operasjonar, setja entitetar ut av spel for kortare eller lengre tidsrom osv.

Tabell 2.4 syner kva for mekanismar ein reknar som høvelege for kvar tenestene. Oversikten syner òg i kva for kommunikasjonslag tenestene bør implementerast.

Service	Mechanism								Layer						
	Encipher-ment	Digital signature	Acces control	Data integrity	Authentication exchange	Traffic padding	Routing control	Notarization	1	2	3	4	5	6	7*
Peer entity authentication	Y	Y	-	-	Y	-	-	-	-	Y	Y	Y	-	-	Y
Data origin authentication	Y	Y	-	-	-	-	-	-	-	Y	Y	Y	-	-	Y
Access control service	-	-	Y	-	-	-	-	-	-	Y	Y	Y	-	-	Y
Connection confidentiality	Y	-	-	-	-	-	Y	-	Y	Y	Y	Y	-	Y	Y
Connectionless confidentiality	Y	-	-	-	-	-	Y	-	-	Y	Y	Y	-	Y	Y
Selective field confidentiality	Y	-	-	-	-	-	-	-	-	-	-	-	-	Y	Y
Traffic flow confidentiality	Y	-	-	-	-	Y	Y	-	Y	-	Y	-	-	-	Y
Connection Integrity with recovery	Y	-	-	Y	-	-	-	-	-	-	-	Y	-	-	Y
Connection integrity without recovery	Y	-	-	Y	-	-	-	-	-	Y	Y	Y	-	-	Y
Selective field connection integrity	Y	-	-	Y	-	-	-	-	-	-	-	-	-	-	Y
Connectionless integrity	Y	Y	-	Y	-	-	-	-	-	Y	Y	Y	-	-	Y
Selective field connectionless integrity	Y	Y	-	Y	-	-	-	-	-	-	-	-	-	-	Y
Non-repudiation Origin	-	Y	-	Y	-	-	-	Y	-	-	-	-	-	-	Y
Non-repudiation. Delivery	-	Y	-	Y	-	-	-	Y	-	-	-	-	-	-	Y
	Y	The mechanism is considered to be appropriate, either on its own or in combination with other mechanisms.							Y	The service should be incorporated in the standards for the layer as a provider option					
	-	The mechanism is considered not to be appropriate							-	Not provided					
		Note – In some instances, the mechanism provides more than is necessary for the relevant service, but could nevertheless be used.							*	It should be noted, with respect to layer 7, that the application process may, itself, provide security services.					

Tabell 2.4 Samanhengen mellom tenester, mekanismar og OSI-lag (17)

2.7 Oppsummering

Dei fire submodellane i OSI drift-og styringsarkitektur er gjennomgått. Funksjonsmodellen definerer og spesifiserer ei rekkje generiske funksjonar for drift og styring av system. Informasjonsmodellen er strengt objektorientert. Objektklassane vert spesifisert med pakkar,

attributtar, operasjonar og notifikasjonar. Kommunikasjonsmodellen nyttar ei spesiell informasjonsteneste, og har sin eigen drift- og styringsprotokoll. Organisasjonsmodellen handterer domene, reglar og mynde. ASN.1 vert nytta i alle spesifikasjonar. Til slutt er OSI sikringstenester og sikringsmekanismar kort gjennomgått.

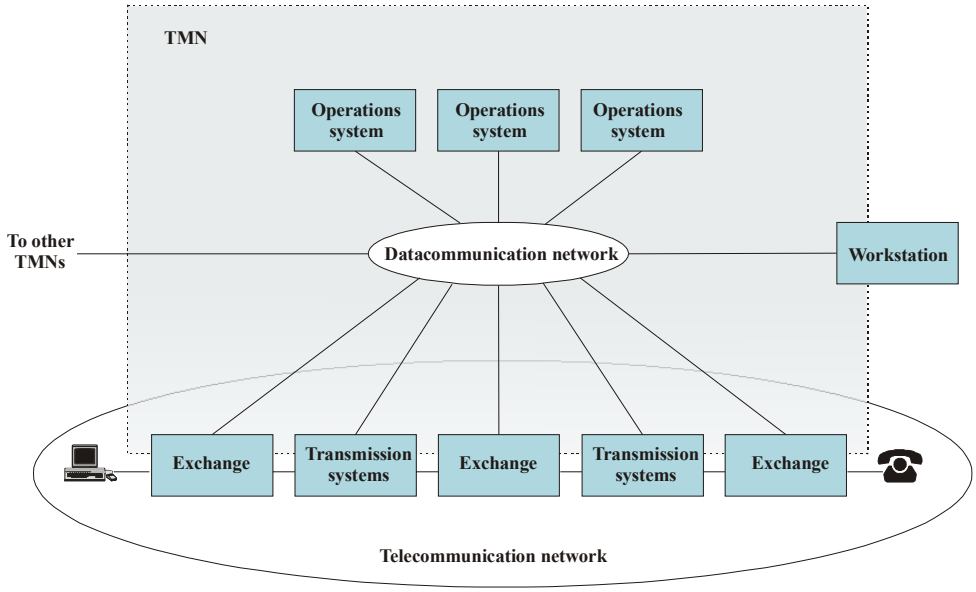
3 TELECOMMUNICATION MANAGEMENT NETWORK (TMN)

TMN er basert på ISO/IEC/ITU-T-standarden som vart gjennomgått i forrige kapittel. Dette kapittelet går særleg inn på område der TMN-standarden inneheld vesentlege tillegg. TMN er spesifisert i ITU-T- rekommendasjonar, i serien M.3000 – M.3599.

3.1 Generelt om TMN

TMN er utvikla for å handtera store heterogene nett, utstyr og tenester. TMN skal operera på utstyr og teknologiar frå mange ulike leverandørar.

Ein kan sjå TMN som eit nett overlagra trafikknettet til telekommunikasjonsoperatøren. TMN skal sørgja for at ein er i stand til å styra og driva eit telekommunikasjonsnett og dei tenestene dette nettet tilbyr. TMN spesifiserer òg kommunikasjonen mellom seg sjølv og telekommunikasjonsnettet/tenestene.



Figur 3.1 Samanhengen mellom TMN og eit telekommunikasjonsnett (18)

3.1.1 Område for drift og styring av telekommunikasjon

Eit område for drift og styring av telekommunikasjon vert definert slik (22):

> *Telecommunications Managed Area is a set of telecommunications resources, logically and/or physically involved with the telecommunications services, that makes it possible to provide, partly or completely, these services to the customers and is chosen to be managed as a whole.*
> *Examples: Switched Data Network*
> *Switched Telephone Network*

Eit drift- og styringsområde kan omfatta alt frå eitt einskild telekommunikasjonsutstyr til komplekse offentlige eller private nett. Avhengig av kompleksitet i nettet, vil kvar bedrift organisera TMN på ulikt vis. Det finns ikkje standardiserte spesifikasjonar for å integrera

styringa av dei ulike områda.

3.1.2 Drift- og styringstenester

Ei rekkje drift- og styringstenester er definert. Det finns ikkje standardiserte spesifikasjonar for å integrera desse tenestene. Tabell 3.1 syner samanhengen mellom ulike område og definerte tenester.

	TMN Users												
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Telecommunication Managed Areas / **Management Services**	Switched Telephone Network	Mobile Communications Network	Switched Data Network	Intelligent Network	CCSS No. 7	N-ISDN	B-ISDN	Dedicated and Reconfigurable Cirquits	TMN	IMT-2000	Access and Terminal Equipment Network	Transport Network	Infrastructure
	1	2	3	4	5	6	7	8	9	10	11	12	13
Customer Administration	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y		
Network Provisioning Management	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Work Force Management	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Tariff, Charging and Accounting Administration	Y	Y	Y	Y		Y	Y	Y		Y			
Qality of Service and Network Performance Administration	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
Traffic Measures and Analysis Administration	Y	Y	Y	Y	Y	Y	Y		Y	Y		Y	
Traffic Management	Y	Y	Y	Y	Y	Y	Y		Y	Y		Y	
Routing and Digit Analysis Administration	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y			
Maintenance Management	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Security Administration	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Logistics Management	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Y	Telecommunication Managed Area in the column needs the Management Service indicated in the row.											

Tabell 3.1 Oversikt over område og tenester for drift og styring (22)

Tenestene vert spesifisert pr område. I desse spesifikasjonane går det fram kva for funksjonalitet og funksjonar som skal nyttast for å realisera tenesta. Funksjonalitet og funksjonar vert gjennomgått i avsnitta 3.2.2 og 3.2.3.

3.1.3 Arkitektur

Ein oversikt over arkitekturen er gitt i (18). TMN-arkitekturen er sett saman av:

- Funksjonell arkitektur; arkitekturen kan delast inn i logiske lag.
- Informasjonsarkitektur, arkitekturen kan delast inn i logiske lag.
- Fysisk arkitektur.

Kommunikasjon vert handsama dels i den funksjonelle og dels i den fysiske arkitekturen.

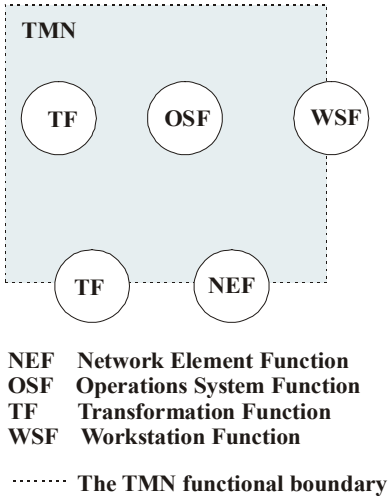
Funksjonar og informasjonsmodellar som er spesifikke for eitt område, kan vera spesifisert i vedkomande område sin rekommendasjonsserie, til dømes i G-serien for transportnett.

3.2 Funksjonar

Det er fire viktige element i den funksjonelle arkitekturen: Funksjonsblokkar, funksjonalitet, drift- og styringsfunksjonar og referansepunkt.

3.2.1 Funksjonsblokkar

Funksjonsblokkane skal gjera det mogleg å utføra TMN drift- og styringsfunksjonar. Funksjonsblokkane er synt i figur 3.2.



Figur 3.2 TMN funksjons-blokkar (18)

- *Operation Systems Function (OSF) block* prosesserer informasjon relatert til monitorering/koordinering/styring av allmenne telekommunikasjonsfunksjonar så vel som av TMN drift-og styringsfunksjonar.
- *Network Element Function (NEF) block* kommuniserer med TMN med det formålet å verta monitorert/styrt. Ein del av blokken ligg utanfor TMN fordi han òg inneheld dei allmenne telekommunikasjonsfunksjonane til nettelementet. Desse er ikkje ein del av TMN, men vert presentert for TMN ved hjelp av NEF.
- *Work Station Function (WSF) block.* Formålet med blokken er å oversetja mellom TMN- entitetar og ikke-TMN entitetar. Ein del av blokken er difor plassert utanfor TMN-grensa. Blokken sørgjer mellom anna for å oversetja til/frå eit brukargrensesnitt. Menneske-maskin-grensesnittet er ikkje del av TMN.
- *Transformation Function (TF) block*. Blokken sørgjer for at ein kan kopla saman entitetar som har innbyrdes inkompatible kommunikasjonsmekanismar, til dømes protokollar og/eller informasjonsmodellar. Blokken kan nyttast kvar som helst; innanfor eit TMN eller på grensa mot andre system. Innanfor eit TMN vil blokken binda saman to funksjonsblokkar som beggje støttar standardiserte, men likevel ulike, kommunikasjonsmekanismar. Når TF vert nytta på grensa mellom to TMN, vil blokken òg i dette tilfellet binda saman to blokkar som beggje støttar standardiserte, men ulike kommunikasjonsmekanismar. Dei to blokkane vil i dette tilfellet liggja i kvart sitt TMN. Når TF vert nytta på grensa mellom eit TMN og eit ikkje-TMN-miljø, vil blokken knyta ein funksjonsblokk som støttar standardiserte kommunikasjonsmekanismar innanfor TMN, saman med ein funksjonell entitet med ikkje-standardiserte kommunikasjonsmekanismar utanfor.

3.2.2 Funksjonalitet

TMN har to former for funksjonalitet:

3.2.2.1 Applikasjonsfunksjonalitet

Management Application Functionality (MAF) representerer funksjonaliteten i ei eller fleire av drift- og styringstenestene som vart synt i tabell 3.1. MAF er implementert i funksjonsblokkane, og ein får dermed fire typar applikasjonsfunksjonalitet:

- *Operations Systems Functionality – Management Application Functionality* (OSF-MAF)
- *Network Element Functionality – Management Application Functionality* (NEF-MAF)
- *Transformation Functionality – Management Application Functionality* (TF-MAF)
- *Work Station Functionality – Management Application Functionality* (WSF-MAF)

MAF er obligatorisk og spesifikk for kvar funksjonsblokk. MAF kan ha manager- eller agentrolle.

3.2.2.2 Støttefunksjonalitet

Støttefunksjonaliteten er i motsetning til applikasjonsfunksjonaliteten, valfri for blokken. Han kan dessutan vera felles for fleire funskjonsblokkar. Døme på slik funksjonalitet er:
- *Data Communication Functionality* (DCF)
- *Workstation Support Functionality*
- *User Interface Support Functionality*
- *Directory System Functionality*
- *Database Functionality*
- *Security Functionality*
- *Message Communication Functionality*

3.2.3 Drift- og styringsfunksjonar

Å utføra ei drift- og styringstenestene vil seia å setja i gang interaksjonar mellom MAFs i ulike funksjonsblokkar ved hjelp av støttefunksjonar. Desse interaksjonane er referert til som TMN *drift-og styringsfunksjonar*. Dei funksjonane som samla støttar interaksjonane til ein einskild MAF vert referert til som eit *funksjonssett*. Funksjonssetta er spesifisert i (25). Funksjonssetta er sett saman til *funksjonssettgrupper*. Desse gruppene kan relaterast til dei fem funksjonsområda som ISO/ITU-T har definert, sjå avsnitta 2.2.2 og 2.3. Prinsippet er forsøkt synt i tabell 3.2, der alle funksjonssettgrupper er lista. Tabellen syner berre nokre få døme på dei svært mange funksjonssetta og funksjonane.

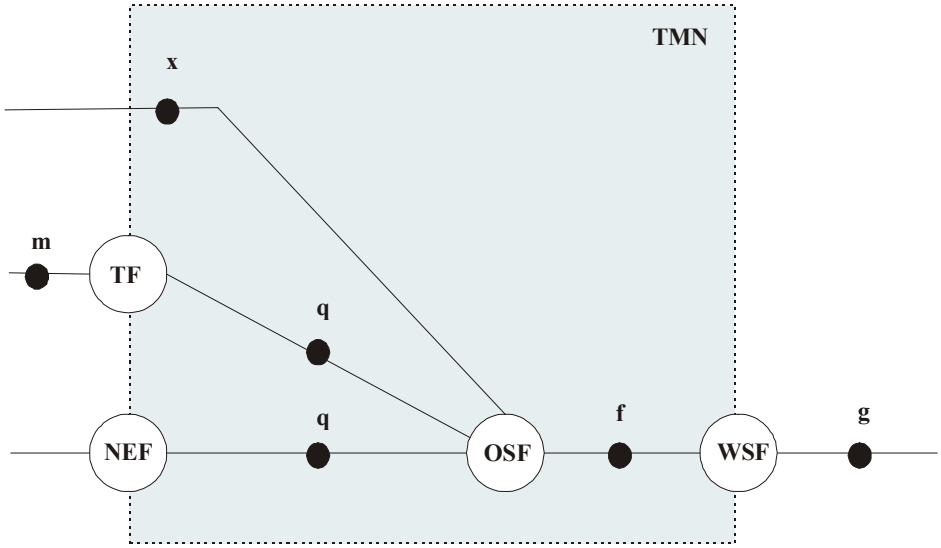
ISO/ITU-T funksjons-område	**TMN Funksjonssettgruppe** (i parantes talet på funksjonssett i denne gruppa)	**TMN Funksjonssett**	**TMN Funksjon**
Feil-handtering	Reliability, Availability and Surviveability (RAS) Quality Assurance (6)		
	Alarm Surveillance (10)		
	Fault Localization (5)		
	Fault Correction (5)		
	Testing (11)		
	Trouble Administration (6)		
Bruks-registrering	Usage Measurement (17)		
	Tarifing/Pricing (8)		
	Collections and Finance (21)		
	Enterprise Control (11)		
Kon-figurering	Network Planning and Engineering (11)		
	Installation (12)		
	Service Planning and Engineering (10)		
	Provisioning (29)	Request for service	Report creation of service resource to customer
			Report deletion of service resource to customer
			Report configuration change of service resource to customer
			Report service state change of service resource to customer
			Request of information about service resource by customer
	Status and Control (8)		
Yting	Performance Quality Assurance (7)		
	Performance Monitoring (10)	Traffic performance monitoring	
		Traffic status	
		Network traffic management policy	
	Performance Management Control (6)	Traffic administration	
		Traffic control	
		Execution of traffic control	
		Detection, counting, storage and reporting	
	Performance Analysis (11)	NE(s) traffic capacity analysis	
		Traffic capacity analysis	
		Customer traffic performance summary	
Datasikring	Prevention (5)		
	Detection (10)		
	Containment and Recovery(16)		
	Security Adminstration (24)		

Tabell 3.2 Oversikt over TMN funksjonssettgrupper med nokre utvalde funksjonssett og funksjonar

3.2.4 Referansepunkt

Omgrepet r*eferansepunkt* er sentralt i TMN. Desse punkta definerer tenestegrenser mellom funksjonsblokkane

Eit referansepunkt gjev eit utsnitt av funksjonaliteten i ein funksjonsblokk ved å syna ei gitt mengd av funksjonane som blokken tilbyr. Dette kan til dømes vera dei operasjonane og notifikasjonane som denne blokken støttar. Dermed referansepunkt brukast til å karakterisera interaksjonane mellom spesifikke par av funksjonsblokkar. Sidan desse interaksjonane dreier seg om informasjonsutveksling mellom funksjonsblokkane, gir referansepunkta eit rammeverk for å spesifisera grensesnitta i TMN. Kvart referansepunkt spesifiserer ein grensesnittkarakteristikk for informasjonsutvekslinga, men definerer ikkje sjølve protokollane.



Figur 3.3 Klassar av referansepunkt i TMN (18)

Figur 3.3 syner eit døme på bruk av dei tre klassane av TMN referansepunkt:
- *q* mellom NEF og OS og mellom TF og OSF.
- *f* mellom OSF og WSF.
- *x* mellom OSFs i to ulike TMNs eller mellom OSF i eit TMN og ein ekvivalent OSF-liknande funksjonalitet i eit anna nett.

I tillegg ser ein to andre klasser av ikkje-TMN referansepunkt:
- *g* mellom ein WSF og brukarar
- *m* mellom TF og entitetar utanfor TMN.

Tabell 3.3 syner mellom kva for blokkar dei ulike referansepunktklassane ligg.

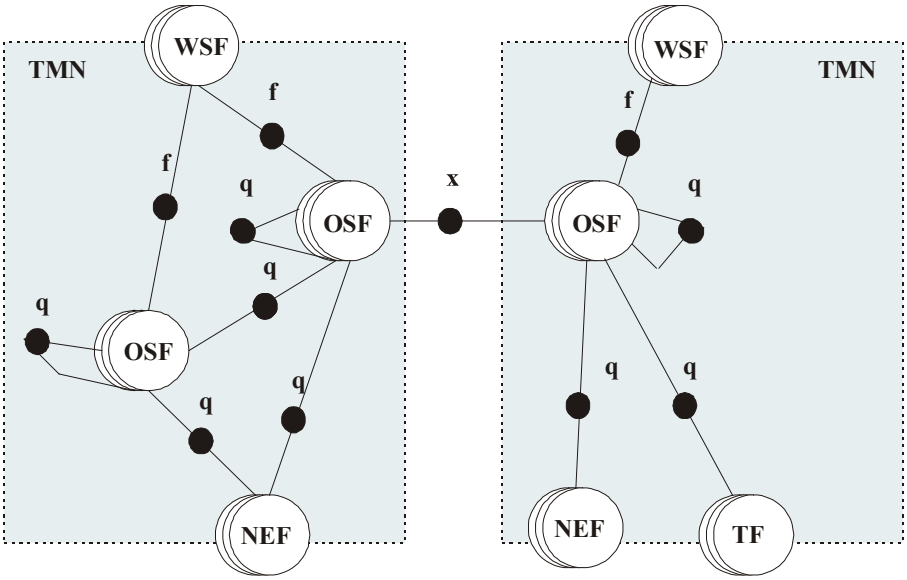
Funksjonsblokk	NEF	OSF	TF	WSF	ikkje-TMN
NEF		*q*	*q*		
OSF	*q*	*q*, *x* [a)]	*q*	*f*	
TF	*q*	*q*	*q*	*f*	*m* [c)]
WSF		*f*	*f*		*g* [b)]
ikkje-TMN			*m* [c)]	*g* [b)]	

a) *x* referansepunkt vert berre nytta når kvar OSF er i ulike TMN.
b) *g* referansepunkt ligg mellom WSF og brukar (menneske)
c) *m* referansepunkt ligg mellom TF og telekommunikasjonsfunksjonalitet

Ein kvar funksjonsblokk kan kommunisera med eit ikkje-TMN referansepunkt. Desse ikke-TMN referansepunkta kan vera standardiserte av andre grupper eller organisasjoner for ulike formål.

Tabell 3.3 Samanhengen mellom funksjonsblokkar og referansepunkt

Figur 3.4 illustrerer referansepunkt mellom funksjonsblokkar.



Figur 3.4 Rreferansepunkt mellom funksjonsblokkar i ein TMN-konfigurasjon (18)

3.2.5 Logiske lag innanfor den funksjonelle arkitekturen

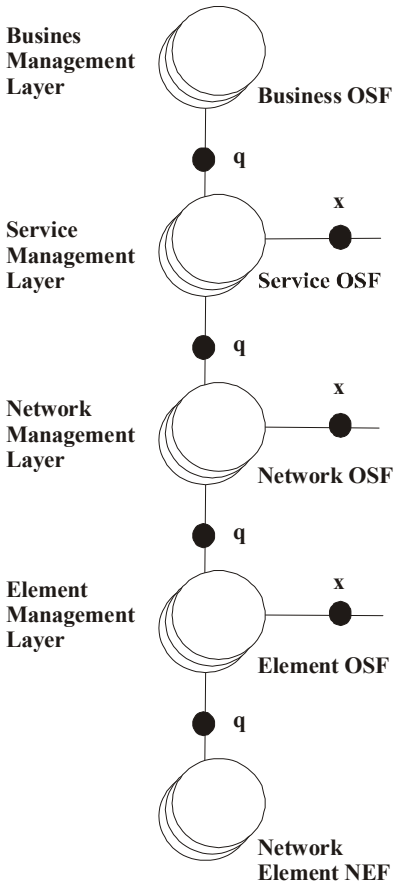
For å handtera kompleksiteten i drift og styring av telekommunikasjon, vert funksjonaliteten delt inn i *logiske lag*. Dei logiske laga reflekterer spesielle drift- og styringsaspekt på ulike abstraksjonsnivå. Desse abstraksjonsnivåa er synt i figur 3.5.

Ei lagdeling av funksjonar medfører ei spesialisering av OSF-blokken. Ei spesialisering basert på ulike abstraksjonsnivå, gir følgjande subklassar av TMN-funksjonalitet:

- *Element*. Element-OSF handterer drift og styring av dei individuelle elementa i eit nett. Elementlaget har tre viktige oppgåver:
  - Styra og koordinera eit subsett av nettelement på individuell basis. OSF støttar

interaksjon med nettelementlaget under og nettlaget over ved å prosessera informasjonen som vert utveksla mellom nett-OSF og den einskilde NEF.
- Styra og koordinera eit subsett av nettelement på kollektiv basis
- Vedlikehalda statistikk, loggar og andre data frå nettelementa i det kontrollerte/styrte området.

- *Network*. Nett-OSF handterer drift og styring av nett og er støtta av elementlaget. Nettlaget styrer eit nett som kan vera lokalisert over eit stort geografisk område. Målet er at dette laget skal kunna gi laget over, tenestelaget, eit teknologisk uavhengig bilete av kommunikasjonsnettet. Laget har fem viktige oppgåver:
  - Styra og koordinera eit heilt nett.
  - Setja parametrar for å framskaffa, modifisera og stoppa tenester til kundane.
  - Vedlikehalda nettet.
  - Vedlikehalda statistikk, loggar og andre data om nettet og samhandla med tenestelaget med omsyn til yting og til bruk av nettet, samt i kva grad nettet er tilgjengeleg.
  - Handtera relasjonane, til dømes konnektiviteten, mellom NEFs.
- *Service*. Teneste-OSF tek seg av tenestene som vert tilbydd av eitt eller fleire nett og vil vanlegvis ivareta rollen som grensesnitt mot kunden. Tenestelaget er ansvarleg for kontraktsaspekta ved tenester som skal leverast kundane, eller som skal vera tilgjengelege for potensielt nye kundar. Hovedfunksjonane til dette laget er å handtera ordrar, innmelde problem og fakturering. Laget har fire viktige oppgåver:
  - Vera første kontaktpunkt (*basic point of contact*) for kunden og andre nettadministrasjonar.
  - Samhandla med andre tenesteleverandørar.
  - Vedlikehalda statistiske data, til dømes om tenestekvalitet.
  - Interaksjonen mellom ulike tenester
- *Business*. Forretnings-OSF er ansvarlig for og koordinerer den totale verksemda; både nett og tenester. I motsetning til nett- og tenestelaget der funksjonane er knytt til optimal utnytting av *eksisterende* telekommunikasjonsressursar, vil dette laget fokusera optimal investering i, og bruk av, *nye* ressursar. Laget har fire viktige oppgåver:
  - Støtta avgjersleprosessar som vedrører optimal investering i, og bruk av, nye telekommunikasjonsressursar.
  - Støtta styring av budsjett relatert til drift og styring.
  - Støtta prosessar relatert til arbeidskraft for drift og styring.
  - Vedlikehalda aggregerte data om den totale verksemda.



Figur 3.5 Logiske lag i TMN

Innan same TMN kommuniserer OSFs innbyrdes innan eit logisk lag, innan to vertikalt påfølgjande lag eller over eit spenn av lag. Kommunikasjonen går gjennom eit *q* referansepunkt. *Mellom ulike* TMNs foregår all OSF-OSF-kommunikasjon gjennom eit *x* referansepunkt.

3.3 Informasjon

3.3.1 Generic Network Information Model (GNIM)

Informasjonsmodellen, GNIM (21), er fullt ut basert på ISO/ITU-T-standarden som vart gjennomgått i avsnitt 2.2.3 og 2.4. Det vert definert generiske objektklassar som vert gruppert i seks *fragment*:

- *Nettfragmentet* omfattar den sentrale objektklassen:
  - *Network*. Objekta er *samlingar* av samankopla logiske eller fysiske telekommunikasjons- eller spesielle drift-og styringsobjekt som er i stand til å utveksla informasjon. Instansane av ein slik objektklasse har felles karakteristikkar som til dømes at dei er eigd av same kunde eller at dei kan assosierast med eit spesifikt tenestenett.
- *Styrt elementfragmentet* omfattar dei sentrale objektklassane:
  - *Managed Element*. Objekta representerer telekommunikasjonsutstyr eller TMN-entitetar som utfører funksjonar innan eit telekommunikasjonsnett. Eit styrt element kommuniserer med manager over eit *q* referansepunkt med formål å verta overvaka eller styrt.
  - *Managed Element Complex*. Formålet med klassen er å gjera det mogleg å referera til meir enn eitt nettelement om gongen.
  - *Equipment*. Objekta representerer dei fysiske komponentane i *Managed Element*-klassen.
  - *Software*. Objekta representerer den logiske informasjonen i *Equipment*-klassen, til dømes program og datatabellar.
- *Termineringspunktfragmentet* omfatter den sentrale objektklassen:
  - *Termination Point*. Objekta representerer termineringspunkta til ein transportentitet, som til dømes eit samband.
- *Svitsjing og transmisjonsfragmentet* omfatter dei sentrale objektklassane:
  - *Circuit End Point Subgroup*. Objekta representerer endepunkta til samlinga av linjer som knyter to ulike sentralar (*exchanges*) saman, og som har dei same eigenskapane.
  - *Pipe*. Objekta representerer entitetar som overfører informasjon mellom to objekt i *Termination Points*-klassen.
- *Krysskoplingsfragmentet* omfattar dei sentrale objektklassane:
  - *Group Termination Point*. Objekta representerer ei gruppe av objekt i *Termination Points*-klassen, og som er slik at dei kan handterast undet eitt.
  - *Cross Connection*. Objekta representerer innbyrdes relasjonar mellom objekt i *Termination Points*-klassen eller i *Group Termination Points*-klassen, til dømes koplinga mellom termineringspunkta til to ledd av eit samband.
  - *Multi-point Cross Connection*. Objekta representerer krysskoplingar av multipunkt-konfigurasjonar.
  - *TP Pool*. Objekta representerer ei gruppe av objekt i *Termination Points*-klassen eller i *Group Termination Points*-klassen, og som er slik at dei kan brukast for eit spesifikt formål, til dømes ruting.
  - *Fabric*. Objekta representerer funksjonar for å handtere krysskoplingar og for å tildela objekt i *Termination Point*-klassen objekt i *Group Termination Point*-klassen og *TP Pool*-klassen.
- *Funksjonsområdefragmentet* omfattar de sentrale objektklassene:
  - *Alarm Severity Assignment Profile*. Objekta er støtteobjekt for drift og styring av alarmar fra dei styrte objekta i nettet.

- *Discriminator*, sjå avsnitt 2.4.2.
- *Log*, sjå avsnitt 2.4.2.
- *Log Record*, sjå avsnitt 2.4.2.

Alle desse objektklassane er definert under superklassen *Top*, sjå avsnitt 2.4.2. Alle er definert med subklassar. GNIM spesifiserer vidare pakkar, attributtar, operasjonar, åtferd, namnebindingar, aksjonar, notifikasjonar og parametrar.

Med utgangspunkt i GNIM er det utarbeidd ei rekkje informasjonsmodellar som er relatert til dei ulike områda for drift og styring av telekommunikasjon. Dette gjeld til dømes for områda Transportnett og ISDN.

3.3.2 Informasjon i eit referansepunkt

Eit subsett, eller eit utsnitt, av informasjonsmodellen vert avbilda til kvart referansepunkt. Utsnittet vil vera relatert til funksjonaliteten i dette referansepunktet. Den ”synlege” informasjonen i eit referansepunkt representerer eit minimum av informasjon som kan spesifiserast for ein funksjonsblokk.

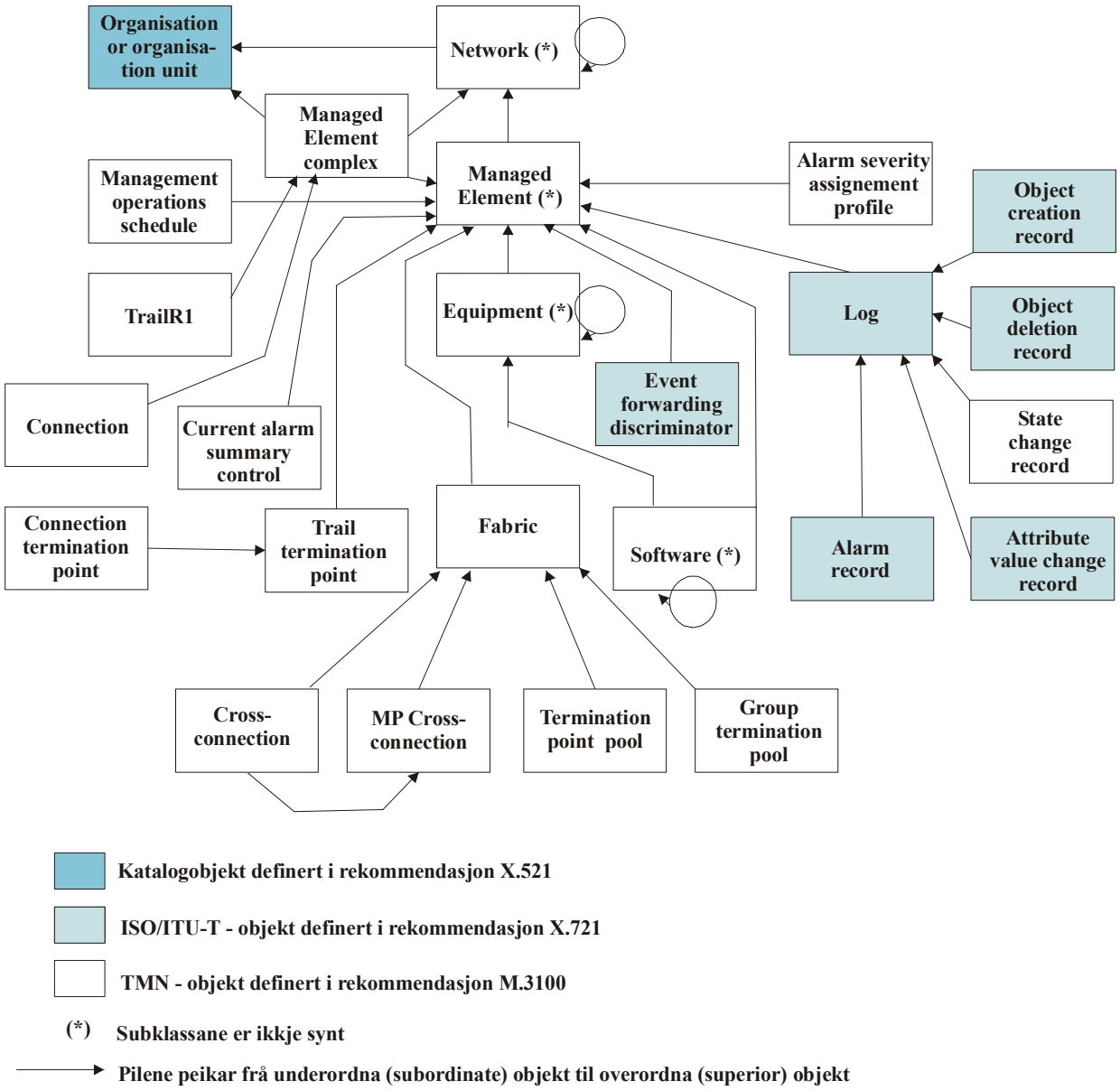
3.3.3 Logiske lag innanfor informasjonsarkitekturen

På same vis som funksjonsmodellen, kan informasjonsmodellen lagdelast. Dette gjer ein gjennom ein serie av utsnitt. Utsnitta er tilpassa abstraksjonsnivået for kvart logisk lag. Kvart utsnitt inneheld informasjon som kan synast eller utvekslast over referansepunkt mellom funksjonsblokkar i dei ulike laga.

3.3.4 Kataloginformasjon

Mykje av informasjonen som trengs for drift og styring, kan hentast frå ei katalogteneste. Dette gjeld til dømes i samhandling på tvers av TMN-domena til ulike nettoperatører eller ulike tenesteleverandørar. Ein kan då oppretta relasjonar mellom katalogobjekta i katalogbasen og de styrte objekta i TMN-basen. Dette medfører at eit styrt objekt i eit TMN vil få eit globalt namn og vera synleg for eit anna TMN.

Namnetreet, sjå avsnitt 2.4.10.3, for GNIM er synt i figur 3.6.



Figur 3.6 Namnetreet for den generiske informasjonsmodellen (21)

3.4 Kommunikasjon

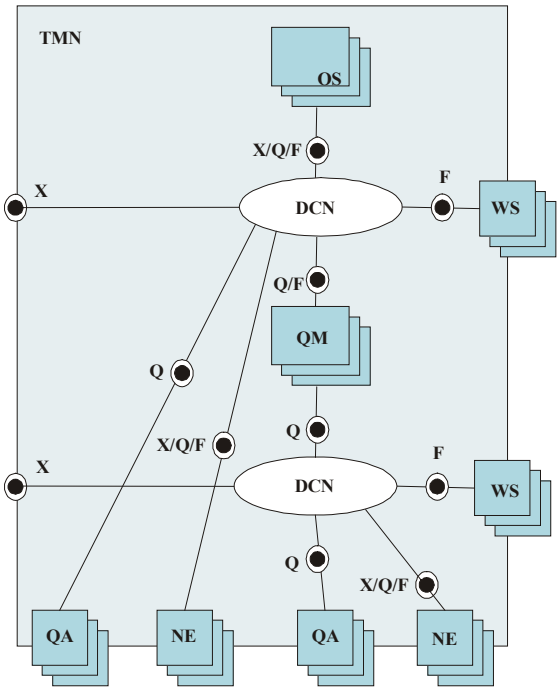
3.4.1 Fysisk arkitektur

Den fysiske arkitekturen er sett saman av:

- *Fysiske blokkar* som har namn etter funksjonsblokkane som vart gjennomgått i avsnitt 3.2.1. Dei kan sjåast som ei fysisk realisering av desse. Til dømes er *Operation System* det systemet som utførerer OSF medan *Data Communication Network* (DCN) utfører *Data Communication Function* (DCF).

- *Grensesnitt* som har namn etter referansepunkta som vart gjennomgått i avsnitt 3.2.4. Dei kan sjåast som ei fysisk realisering av desse. For grensesnitta nyttar ein store bokstavar, til dømes: Grensesnitt *X* i referansepunkt *x*.

TMN kan implementerast i ei mengd ulike fysiske konfigurasjonar. Eit døme er synt i figur 3.7. Dei fysiske blokkane kan ha andre typar grensesnitt og funksjonalitet enn det som vert omtalt i dette avsnittet. Dette ligg i så fall utanfor TMN.



Figur 3.7 *Døme på fysisk arkitektur for TNM; fysiske blokkar og grensesnitta mellom dei (19)*

3.4.1.1 Fysiske blokkar

Følgjande fysiske blokkar er definert:

- *Operations System* (OS) er det systemet som utfører OSF. OS kan òg innehalda QA og WSF.
- *Q-Adapter* (QA) koplar NE/OS-liknande fysiske blokkar med ikkje-TMN grensesnitt til *Q*-grensesnittet.
- *X-Adapter* (XA) koplar ikkje-TMN fysiske entitetar med ikkje-TMN kommunikasjonsmekanismar utanfor TMN til OS.
- *Q-Mediation* (QM) støttar koplingar mellom ikkje-kompatible entitetar innanfor eit TMN.
- *X-Mediation* (XM) støttar koplingar mellom to ikkje-kompatible OS i to ulike TMN.
- *Network Element* (NE) omfattar telekommunikasjonsutstyr (eventuelt grupper/deler av dette) som utfører NEFs, sjå avsnitt 3.2.1. NE kan i utgangspunktet innehalda alle funksjonsblokkane og har eit eller fleire *Q*-grensesnitt. NE kan òg ha *F*- og *W*-grensesnitt.
- *Work Station* (WS) er systemet som utfører WSF, med mindre ein av dei andre blokkane innkorporerer WSF.
- *Data Communicaiton Network* (DCN) sørgjer for stiar for informasjonsflyt mellom dei fysiske blokkane i TMN. Ein kan sjå DCN som ei samling ressursar som støttar informasjonsoverføringa mellom dei distribuerte TMN-komponentane. DCN kan vera sett saman av ei rekkje individuelle, men samankopla, subnett av ulike slag. DCN er teknologiuavhengig og kan nytta ein eller fleire transmisjonsteknologiar.

Tabell 3.4 syner samanhengen mellom funksjonsblokkar og fysiske blokkar.

Funksjonsblokk / Fysisk blokk	NEF	OSF	TF	WSF
NE	Obligatorisk	Valfri	Valfri	Valfri **)
OS		Obligatorisk	Valfri	Valfri
QA, XA, QM, XM *)			Obligatorisk	
WSF				Obligatorisk

*) Funksjonsblokken *Transformation Function* (TF), sjå avsnitt 3.2.1, syter for konvertering mellom ulike protokollar og ulike format for informasjonsutveksling. I *q*- og *x*-referansepunkta er det to typar transformasjon:
- Tilpassing (*adaption*) mellom ikkje-TMN-entitetar og TMN-entitetar.
- Mekling (*mediation*) mellom ikkje-kompatible TMN-entitetar.

Dette gir opphav til fire ulike fysiske blokkar: Q-Adapter (QA), X-Adapter (XA), Q-Mediation (QM) og X-Mediation (XM):

	Tilpassing	Mekling
Q-grensesnitt (intra-TMN)	QA	QM
X-grensesnitt (inter-TMN)	XA	XM

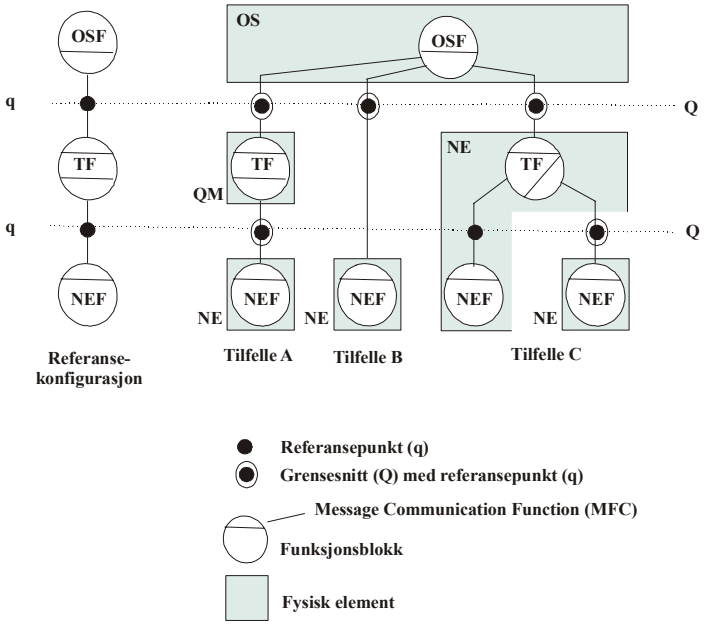
**) Dersom WSF skal implenterast må òg OSF vera tilstades. Dette vil seia at WSF må gå via OSF. Det lokale menneske-maskingrensesnittet er utanfor TMN.

Tabell 3.4 Samanhengen mellom funksjonsblokkar og fysiske blokkar

3.4.1.2 Grensesnitt

I dette avsnittet vert nokre døme på bruka av grensesnitta i ulike konfigurasjonar gjennomgått:

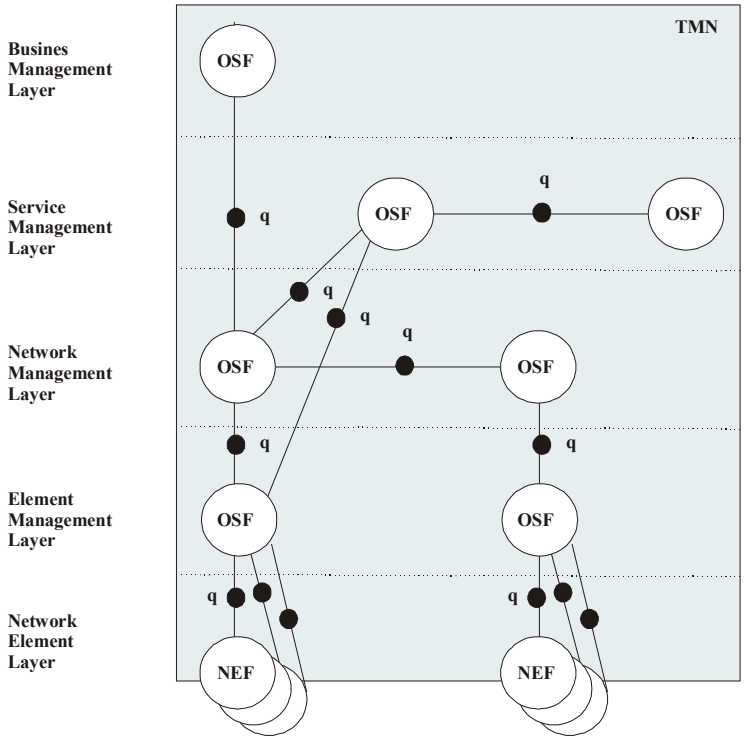
- *Q-grensesnitt mellom OS og NE*. Figur 3.8 syner korleis ein kan oppnå interaksjon mellom OS og NE i ulike fysiske konfigurasjonar. DCN er ikkje synt direkte.
  - Tilfelle A) syner eit NE som via eit *Q*-grensesnitt er kopla til ekstern QM. QM utfører TF for å konvertera mellom dei respektive *Q*-grensesnitta til NE og OS.
  - Tilfelle B) syner eit NE som er kopla direkte til OS via eit *Q*-grensesnitt.
  - Tilfelle C) syner eit NE med intern TF. NE er kopla til OS via *Q*-grensesnitt til OS. Eit anna NE er kopla til over eit felles *Q*-grensesnitt.



Figur 3.8 Samanhengen mellom grensesnitt, referansepunkt, funksjonsblokkar og fysiske blokkar (19)

OSF vil typisk ha MAF i managerrolle, TD vil ha MAF i agentrolle mot OSF og i managerrolle mot NEF. NEF vil ha MAF agentrolle mot OSF og TF, og kan dessutan ha MAF i managerrolle mot andre NEFs.

- *Q-grensesnitt mellom to OSFs innan same TMN*. Slik interaksjon kan som før nemnt forekoma innan eit logisk lag, mellom vertikalt påfølgjande lag eller over eit spenn av lag. Dette er synt i figur 3.9. Figuren syner referansepunkt mellom funksjonsblokkar. I ein fysisk konfigurasjon vil *Q*-grensesnitt kunna forekoma i desse referansepunkta.



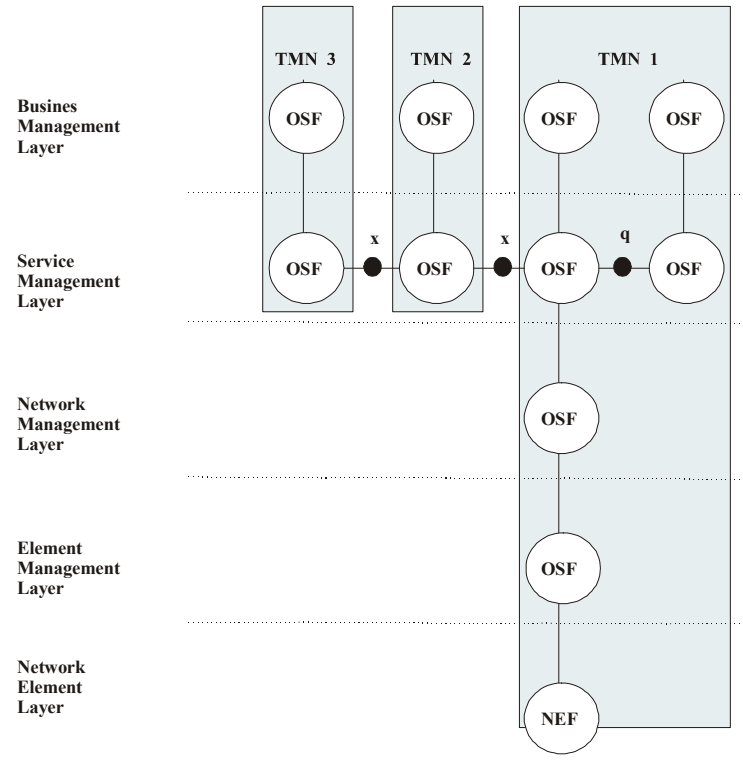
Figur 3.9 Døme på intra-TMN interaksjonar over q referansepunkt (19)

- *X-grensesnitt mellom to OS i ulike TMN*. Mellom to TMN må grensesnittet ta omsyn til både inter-administrative applikasjonar og kommersielle tenester. I *X*-grensesnittet kan det difor vera naudsynt med protokollar og informasjonsmodellar som er ulike eller som kjem i tillegg til dei som støttar *Q*- og *F*-grensesnittet. Grensesnittet kan dermed variera.

  Administrativt vil grensesnittet vera avhengig av krav som følgjer av geografiske og juridiske grenser:
  - Mellom offentlege teleoperatørar.
  - Intra-nasjonalt; mellom offentlege og private operatørar, mellom operatørar og kundar.
  - Internasjonalt.

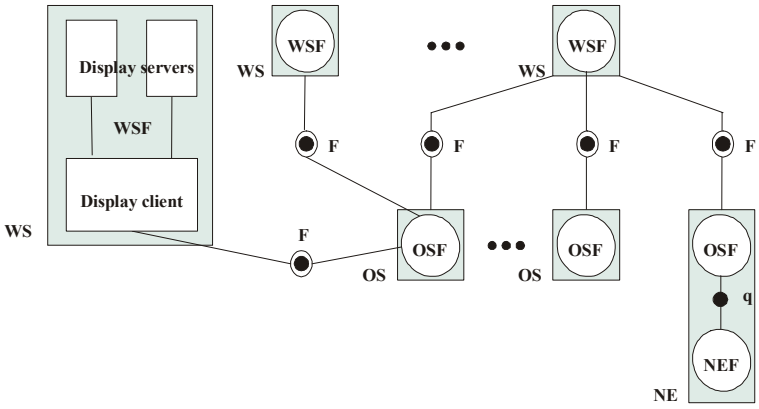
  Ulike TMN-hierarki kan samarbeida og gripa inn i kvarande av mange årsaker:
  - For å styra naudsynt samarbeid for å realisera verdiaukande tenester.
  - For å styra separate geografiske/funksjonelle TMNs som eitt einskild TMN.
  - For å realisera ende-til-ende-samband/tenester.

  Dette er synt i figur 3.10. Ein teleoperatør samarbeider med andre TMNs over *x*-referansepunkt. Interaksjonen foregår i dette tilfellet på tenestelaget, men kan i prinsippet foregå på alle logiske lag. Interaksjonen mot andre TMNs vert støtta gjennom TMN-interne interaksjonar over *q*-referansepunkt.



Figur 3.10 Døme på inter-TMN interaksjonar (19)

- *F-grensesnitt mellom OS og WS*. Figur 3.11 syner nokre døme på konfigerering av WS og OS. Figuren syner og eit døme på distribuert WSF. Ein ser òg eit tilfelle av at WS kommuniserer direkte NE (som då må innehalda OSF).



Figur 3.11 Døme på F-grensesnitt-konfigurasjon (23)

Målet for grensesnittspesifikasjonane er å sikra at dei ulike komponentane som er samankopla er kompatible uavhengig av type og leverandør. Dette krev kompatible kommunikasjon-protokollar og ein kompatibel metode for å representera informasjonen i meldingar.

3.4.2 Protokollar

Grensesnitta definerer protokollar, samt meldingane som protokollane skal overføra.

Transaksjonsorienterte grensesnitt er basert på eit objektorientert syn på kommunikasjon, og alle meldingane som går over desse grensesnitta, handlar difor om å manipulera objekta. Kvart objekt er definert med eit sett lovlege operasjonar. Det finns generiske meldingar som kan nyttast for mange objektklasser. I tillegg finns meldingar som er spesifikke for ein objektklasse.

DCN skal så langt mogleg vera i tråd med OSI referansemodell. Innan eit TMN vil kommunikasjonslinjene variera sterkt. Dei kan vera dedikerte, pakkesvitsja, ISDN, LAN, PSTN osv. Eit kvart nett eller ei blanding av nett kan i utgangspunktet nyttast til å overføra TMN-meldingar. Det er definert protokollprofilar for *X*- og *Q*-grensesnitta. Ein er ikkje ferdig med standardiseringsarbeidet for *F*-grensesnittet.

Val av protokollar på lågare lag vil vera avhengig av den fysiske konfigurasjonen. På høgare lag vil valet vera avhengig av om det dreier seg om transaksjonsorienterte tenester, filtenester eller katalogtenester. Lågare lags protokollprofilar for grensesnitta *X* og *Q* er spesifisert i (27), høgare lags i (28).

3.4.2.1 Lågare lags protokollar for *X*- og *Q*-grensesnittet

Tabell 3.5 syner protokollprofilane som er spesifisert for lag 1 – 4 i OSI referansemodell.

Protokoll	**Merknad**
CONS*)1	Pakkegrensesnitt med tilknyting (connection mode), nyttar X.25
CONS2	Pakkegrensesnitt med tilknyting, nyttar X.31 og ein ISDN D-kanal
CONS3	Pakkegrensesnitt med tilknyting, nyttar X.31 og ein ISDN B-kanal
CONS5	Grensesnitt med tilknyting, nyttar Signalling System No.7 MTP*) og SCCP*).
CONS6	Pakkegrensesnitt med tilknyting, nyttar X.25 over LAN
CLNS*)1	Grensesnitt utan tilknyting (connectionless mode), nyttar ISO/IEC 8802-2 type LAN med CSMA*)/CD*)
CLNS2	Grensesnitt utan tilknyting, nyttar ISO CLNP*) over ein tilknytingsorientert X.25-protokoll (brukt i WAN)
CLNS3	Grensesnitt utan tilknyting, nyttar ISO CLNP over ISDN B-kanal
RFC1006 TCP/IP	Sørgjer for OSI Transportklasse 0 over Internet TCP
*) CONS CLNS MTP SCCP CSMA CD CLNP	Connection mode Network layer Service Connectionless mode Network layer Service Message Transfer Part Signalling Connection Control Part Carrier Sense Multippel Access Collision Detection Connectionless mode Network Layer Protocol

Tabell 3.5 Oversikt over lågare lags protokollprofilar for X- og Q-grensesnittet

Appendix B syner protokollstakken for desse protokollprofilane.

3.4.2.2 Høgare lags protokollar for *X*- og *Q*-grensesnittet

Det vert spesifisert fire protokollprofilar for lag 5-7 i OSI referansemodell:

- *For interaktive tenester*, der ein på sesjonslaget nyttar OSI *Session Protocol* (SP), på presentasjonslaget nyttar OSI *Presentation Protocol* (PP) og på applikasjonslaget, med utgangspunkt i CMISE, nyttar protokollane CMIP, ACSE og ROSE slik dette vart gjennomgått i avsnitt 2.5. For interdomenekommunikasjon kan dessutan CORBA nyttast, og

ein arbeider med spesifikasjonar for bruk av CORBA i intradomene-kommunikasjon. CORBA vert gjennomgått i avsnitt 5.4.
- *For filorienterte tenester,* der ein på sesjons- og presentasjonslaget nyttar OSI SP/PP, men på applikasjonslaget nyttar ISO-protokollen *File Transfer, Access and Management* (FTAM) og ACSE.
- *For katalogtenester,* der ein på sesjons- og presentasjonslaget nyttar OSI SP/PP, og på applikasjonslaget nyttar OSI katalogprotokollane *Directory Access Protocol* (DAP) og *Directory System Protocol* (DSP) (8) samt ACSE og ROSE.
- *CORBA*-basert protokollprofil er spesifisert for *X*-grensesnittet. For *Q*-grensesnittet er tilsvarande ikkje ferdig.

Ein ser at høgare lags protokollprofilar er ulike på applikasjonslaget. Dette vil seia at TMN ikkje er avgrensa til CMIP. Appendix B syner protokollstakken for desse protokollprofilane.

3.5 Datasikring

Som synt i tabell 3.2, er det definert fire funksjonssettgrupper for å handtera datasikringa i kommunikasjonsnettet som TMN styrer. Desse funksjonane handterer tenestene og mekanismane som vart gjennomgått i avsnitt 2.6. I tillegg vert det definert nye mekanismar for til dømes revisjonar, gjenvinning og deteksjon av visse hendingar.

I resten av avsnittet vil ein sjå på korleis sjølve TMN er sikra særskilt. TMN nyttar dei generelle OSI-tenestene/mekanismane. Infrastrukturen for datasikring i eit TMN må ta omsyn til pålegg frå styresmakter, reguleringar, kontraktar og avtalar. Dette kan til dømes dreia seg om kva for sikringstenester som skal som skal vera obligatoriske, kva for mekanismar som skal nyttast osv. (20) gir ein oversikt over datasikringa i TMN.

3.5.1 Trugsmål

Trugsmåla mot TMN vert delt inn i tre typar:
- *Uhell*, som er trugsmål uten "ondsinna" intensjon.
- *Administrasjon*, som er trugsmål som skuldast mangel ved sikringsdministrasjonen.
- *Intensjon*, som er trugsmål der ein "ondsinna" entitet angrip enten kommunikasjonen i seg sjølv eller ein nettverksressurs.

Trugsmåla kan rettast mot:
- Konfidensialiteten til lagra og overført informasjon.
- Dataintegriteten i lagra og overført informasjon.
- Ansvaret (accountability) ein kvar entitet har for inititerte aksjonar.
- Tilgjenget alle lovlege entitetar skal ha til TMN-fasilitetane.

Tabell 3.6 syner kva dei ulike intensjonstrugsmåla kan skada.

Threat	Confidentiality	Data Integrity	Accountability	Availability
Masquerade (spoofing)	x	x	x	x
Eavesdropping	x			
Unauthorized access	x	x	x	x
Loss or corruption of information (transferred)		x		x
Repudiation			x	
Forgery		x	x	
Denial of service				x

Tabell 3.6 Trugsmål og mål (20)

3.5.2 Sikringsdomene

Eit sikringsdomene er definert til å vera eit sett entitetar og partar som er underlagt *eitt* felles regelsett for datasikring med *ein* administrasjon. Eit sikringsdomene *kan* vera samanfallande med eit TMN, men det vil ikkje alltid vera slik. Eit stort TMN kan vera sett saman av fleire drift- og styringssystem, og det kan då vera aktuelt å dela det inn i fleire sikringsdomene. Det vil difor vera meir høveleg å sjå eit sikringsdomene som samanfallande med eitt einskild eller eit sett av OSF-domene.

3.5.3 Sikring i grensesnitt

3.5.3.1 Sikring av *X*-grensesnittet

Det vert ikkje lagt spesielle føringar for sikringsmekanismane i protokollane som er nytta på lågare lag, sjå appendix B.

På høgare lag er støtte til autentiserings- og tilgangskontrolltenestene obligatorisk ved bruk av interaktive tenester. Autentiseringstenesta skal nytta *Authetication Functional Unit* (AFU) som er spesifisert i ACSE. Tilgangskontrolltenesta skal nytta tilgangskontrollparameteren som er definert i CMIP. Dei andre sikringstenestene er valfrie. Einskilde applikasjonar kan likevel medføra krav om at ei eller fleire av dei valfrie sikringstenestene *skal* brukast. (24) skildrar korleis sikringstenester og mekanismar *bør* brukast på *X*-grensesnittet. Her vert òg generelle krav til sikring i dette grensesnittet drøfta.

Ved bruk av filorienterte tenester er autentiseringstenesta obligatorisk, og denne skal òg i dette tilfellet nytta AFU. Mykje er førebels uavklart vedrørande sikring av FTAM-tenester i TMN.

Ved katalogtenester vert det synt til det generelle rammeverket for datasikring i OSI katalogtenester.

3.5.3.2 Sikring av *Q*- og *F*-grensesnitta

Alle sikringstenester er valfrie. (23) listar krav som bør ivaretakast for *F*-grensesnittet.

3.6 Oppsummering

Arkitekturen for TMN er gjennomgått. TMN er med omsyn til funksjonar, informasjonsmodell, kommunikasjon og datasikring fullt ut basert på ISO/ITU-T-standarden for drift og styring. TMN er logisk delt inn i funksjonsblokkar med tilhøyrande funksjonalitet som er konkretisert i form av drift-og styringsfunksjonar. Mellom funksjonsblokkane vert det definert referansepunkt. Funksjonsblokkar og referansepunkt vert realisert som fysiske blokkar med grensesnitt mellom. For kvar type grensesnitt finns ulike protokollprofilar. TMN informasjonsbasar vert bygd opp på bakgrunn av ein generisk modell. Basen kan integrerast med OSI kataloginformasjonsbase. I seinare tid har TMN-spesifikasjonane vorte mykje påverka av dei nye arkitekturane som vert gjennomgått i kapittel 5.

4 SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

I dette kapittelet vert nemninga *SNMP* nytta som samleomgrep for protokollen så vel som for informasjonen og tenestene som er knytt til denne.

SNMP er spesifisert i *Requests for Comments* (RFCs). Arbeidet med desse spesifikasjonane er som for andre Internett-spesifikasjonar, styrt av *Internet Architecture Board* (IAB) gjennom IETF og *Internet Research Task Force* (IRTF). I motsetning til ITU-T, som hovudsakleg er driven av aktørar innan transportnett, er IETF i stor grad driven av utstyrleverandørane.

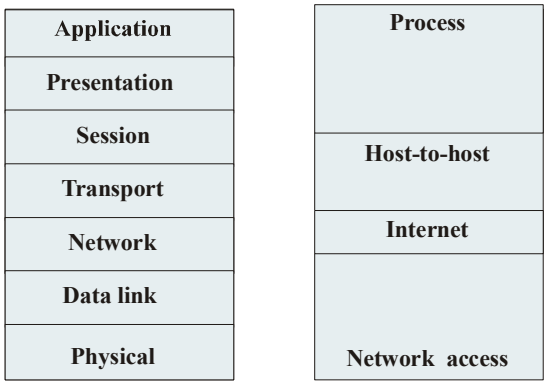
Dei sentrale spesifikasjonene for SNMPv1 har status *Internet Standard*, medan dei tilsvarende for SNMPv2 og SNMPv3 stort sett har status *Draft Standard* eller *Proposed Standard.* Desse verkar uferdige og til dels inkonsekvente og inkonsistente.

Omgrep og nemningar som er nytta i spesifikasjonane for SNMP har sjeldan gjennomgåande formelle definisjonar slik ein er van til frå ISO/ITU-spesifikasjonar. Dermed vert eitt og same omgrep nytta ulikt i dei ulike RFCs for SNMP. Til dømes vert omgrepet *agent* nytta for ein applikasjon i agentrolle så vel som for sjølve det fysiske nettelementet der ein slik applikasjonen er implementert. Omgrepet manager vert på same måte nytta både for ein applikasjonen i managerrolle og for den fysiske arbeidsstasjonen til operatøren.

SNMP er basis for dei fleste drift- og styringsimplementasjonar i IP-nett. Dette gjeld først og fremst LAN, intranett og Internett.

4.1 Generelt om TCP/IP-kommunikasjonsarkitektur

I tillegg til OSI referansemodell, som kort vart presentert i avsnitt 2.1, er det TCP/IP-arkitekturen som har dominert utviklinga av interoperable kommunikasjonsstandardar. Denne protokollfamilien innheld ei rekkje protokollar.
Forholdet mellom kommunikasjonslaga i OSI og TCP/IP kommunikasjonsarkitektur er synt i figur 4.1.



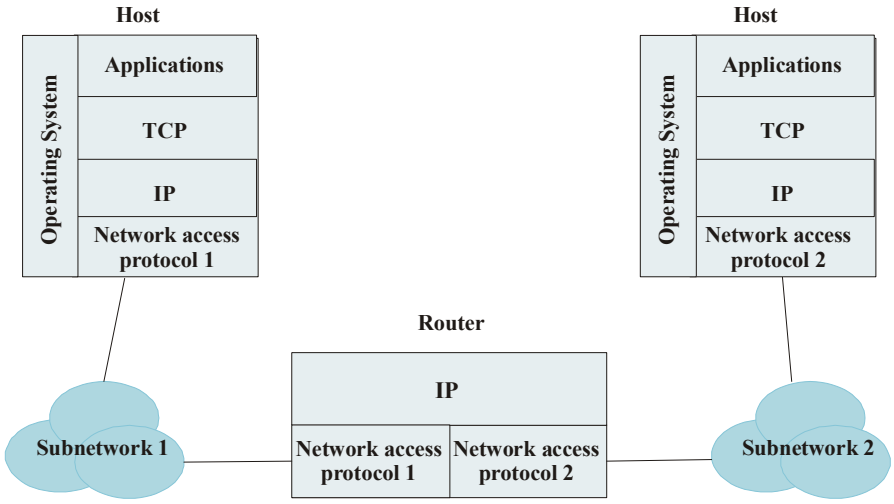
Figur 4.1 *Tilhøvet mellom OSI og TCP/IP kommunikasjonslag (3)*

Arkitekturen kan delast inn i fire lag:

- *Network Access layer* handterer datautveksling mellom eit endesystem, til dømes vertsmaskin eller arbeidsstasjon, og nettet dette systemet er tilknytt. Kva type programvare som er nytta på dette laget avheng av nettype. Nettet kan vera pakkesvitsja, til dømes X.25, eller det kan vera eit lokalnett, til dømes Ethernett. Dette laget handterer tilgang til nettet samt ruting gjennom det.
- *Internet layer* handterer datautveksling mellom endesystem når desse er tilknytt *ulike* nett. Laget har prosedyrar som mogleggjer datautveksling over mange nett. På dette laget vert IP

nytta for å ruta data gjennom multiple nett. Protokollen er implementert i endesystema så vel som i ruterane. (Ein ruter er ein prosessor som som knyter saman to nett. Primæroppgåva er å vidaresenda data frå eitt nett til eit anna på ruta frå kjelde til destinasjon.)

- *Host-to-host layer* handterer ende til ende-transporten. På dette laget vert oftast *Transmission Control Protocol* (TCP) nytta for å sikra at data kjem fram til destinasjonsapplikasjon på korrekt vis.
- *Process layer* inneheld logikk for å støtta ulike brukarapplikasjonar, som til dømes filoverføring og epost. Viktige protokollar på dette laget er *Simple Mail Transfer Protocol* (SMTP), *File Transfer Protocol* (FTP), TELNET og SNMP.



Figur 4.2 Kommunikasjon over TCP/IP (51)

Som eit alternativ til TCP finns UDP, som tilbyr ei tilkoplingsfri teneste med eit minimum av mekanismar. UDP garanterer korkje at data vert levert, at sekvensane kjem i korrekt rekkefølgje eller vern mot duplikat. SNMP nyttar vanlegvis UDP som transportprotokoll.

4.2 Modell for SNMP drift og styring

I løpet av 90-talet har bruken av SNMP eksplodert. Årsaka til dette er at modellen er enkel. Informasjonsbasen tillet berre skalarar og todimensjonale såkalla *konseptuelle tabellar*, medan protokollen grovt sett tillet berre fire operasjonar på denne informasjonen.

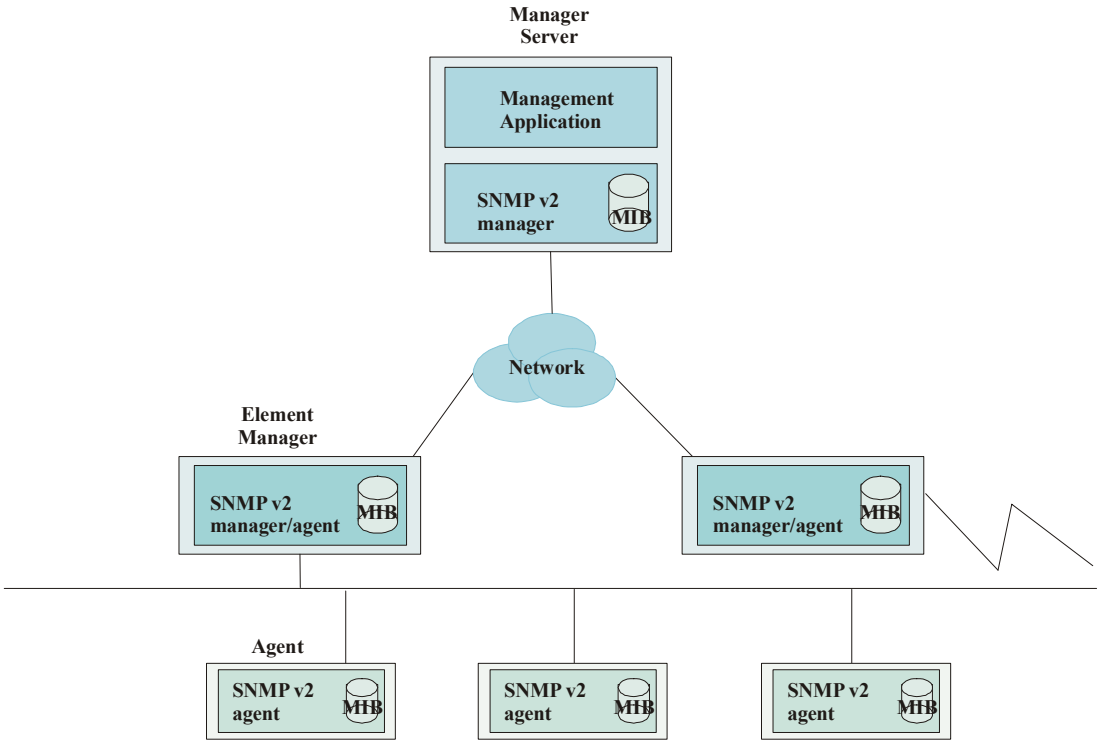
Rundt 1994 starta ein arbeidet med SNMPv2. Arbeidet var motivert ut frå dei svært mangelfulle sikringsmekanismane i SNMPv1. Arbeidet for å bøta på dette mislukkast. Dette er årsak til at:

- SNMPv2 ikkje byr på noko grunnleggjande nytt i høve til SNMPv1.
- Arbeidet med SNMPv3 vart starta i 1998 for å løysa problema rundt datasikring i SNMP-basert drift- og styring. SNMPv3 baserer seg på ein ny arkitektur og innheld spesifikasjonar for datasikring.

Mykje tyder på at det framleis er SNMPv1 som er i praktisk bruk. Vidare ser det ut til at protokollen først og fremst er nytta for overvaking, og i mindre grad til styring av nett, som til dømes konfigurering av nettelement.

SNMP-arkitekturen inneheld korkje funksjons- eller organisasjonsmodell. Arkitekturen er sett saman av fire nøkkelelement (2) for drift og styring:
- Stasjon (*Management Station*).
- Agent (*Management Agent*).
- Informasjonsbase (*Management Informatuon Base,* MIB).
- Protokoll (*Management Protocol*).

Arkitekturen for SNMPv2 skil seg ikkje grunnleggjande frå arkitekturen for SNMPv1. Rammeverket for SNMPv2 finn ein i (34). Den nye arkitekturen for SNMPv3 vert gjennomgått i eit seinare avsnitt.



Figur 4.3 Desentralisert SNMP-konfigurasjon (51)

SNMP er i utgangspunktet basert på *sentralisert* infrastruktur. Figur 4.3 syner eit døme på meir desentralisert hierarkisk infrastruktur, noko som òg er mogleg. I dette oppsettet har ein modular (*element managers*) som fungerer både som manager og som agent. På vegne av manager kan desse modulane monitorera og styra agentane dei har ansvar for. SNMPv2 gir betre støtte for slike løysingar enn SNMPv1.

4.2.1 Stasjon

Stasjonen er grensesnittet mellom brukaren (menneske) og drift- og styringssystemet. Som eit minimum vil stasjonen ha:
- Drift- og styringsapplikasjonar for funksjonar som data-analyse, feilhandtering osv.
- Eit brukargrensesnitt for å overvaka og styra nettet.
- Funksjonalitet for å oversetja brukarkommandoar til reell overvaking og styring av nettelementa.
- Ein database med informasjon som er ekstrahert frå dei ulike informasjonsbasane i dei styrte

entitane i nettet.

Merk at berre dei to siste punkta er standardiserte.

4.2.2 Agent

Ein agent er ein applikasjon i eit styrt nettelement, til dømes vertsmaskin, bru, rutar eller hub. Agenten svarar på forespørslar frå stasjonen ved å utføra operasjonar. Agenten kan òg asynkront levera informasjon til stasjonen. Agenten har tilgang til ein informasjonsbase, MIB. En skil mellom to typar agenter:

- *Management agent* som er implementert i eit nettelement som støttar TCP/IP protokollfamilie.
- *Proxy agent* som tillet monitorering og styring av nettelement som i utgangspunktet ikkje er tilgjengelege for ovannemnte protokollar, eller som ein av andre årsaker ikkje ynskjer å implementera SNMP i. Proxy-agenten inneheld ein funksjon for protokollkonvertering.

4.2.3 Informasjonsbase

Drift- og styringsinformasjonen kan sjåast som ei samling av styrte objekt. Strukturen for denne informasjonen, SMI, spesifiserer korleis objekttypane skal definerast og finns i to versjonar: SMI for SNMPv1 (SMIv1) er definert i (29) og SMI for SNMPv2 (SMIv2) er definert i (44).

Spesifikasjonen av sjølve informasjonsbasane med definisjon av dei konkrete objekttypane, finns i tre versjonar: For SNMPv1 vart først MIB-I definert (30). Noko seinare kom MIB-II (33); framleis for same SNMP-versjon. MIB-II er standard for SNMPv1, og den opprinnelege MIB-I vert difor ikkje omtalt vidare i denne rapporten. For SNMPv2 er ein ny MIB definert i (37).

4.2.4 Protokoll

Stasjonen og agenten utvekslar informasjon ved hjelp av protokollen SNMP. Utvekslinga skjer i form av protokollmeldingar. Monitorering av nettet vert utført først og fremst ved *polling*. Det finns eit avgrensa utval av meldingar. Meldingar frå agenten vert kalla *traps* i SNMPv1 og *notifications* i SNMPv2. Protokollen er ein enkel *request/response*-type-protokoll og har følgjande nøkkelfunksjonar:

- *Get*. Set stasjonen i stand til å henta fram verdiane til objekta i agenten sin MIB.
- *Set*. Set stasjonen i stand til å setja verdiane til objekta i agenten sin MIB.
- *Trap*. Set agenten i stand til å melda frå til stasjonen ved viktige hendingar.

Protokollen finns i tre versjonar, SNMPv1 definert i (31), SNMPv2 definert i (35) og SNMPv3 definert i (42), (43), (39) og (40).

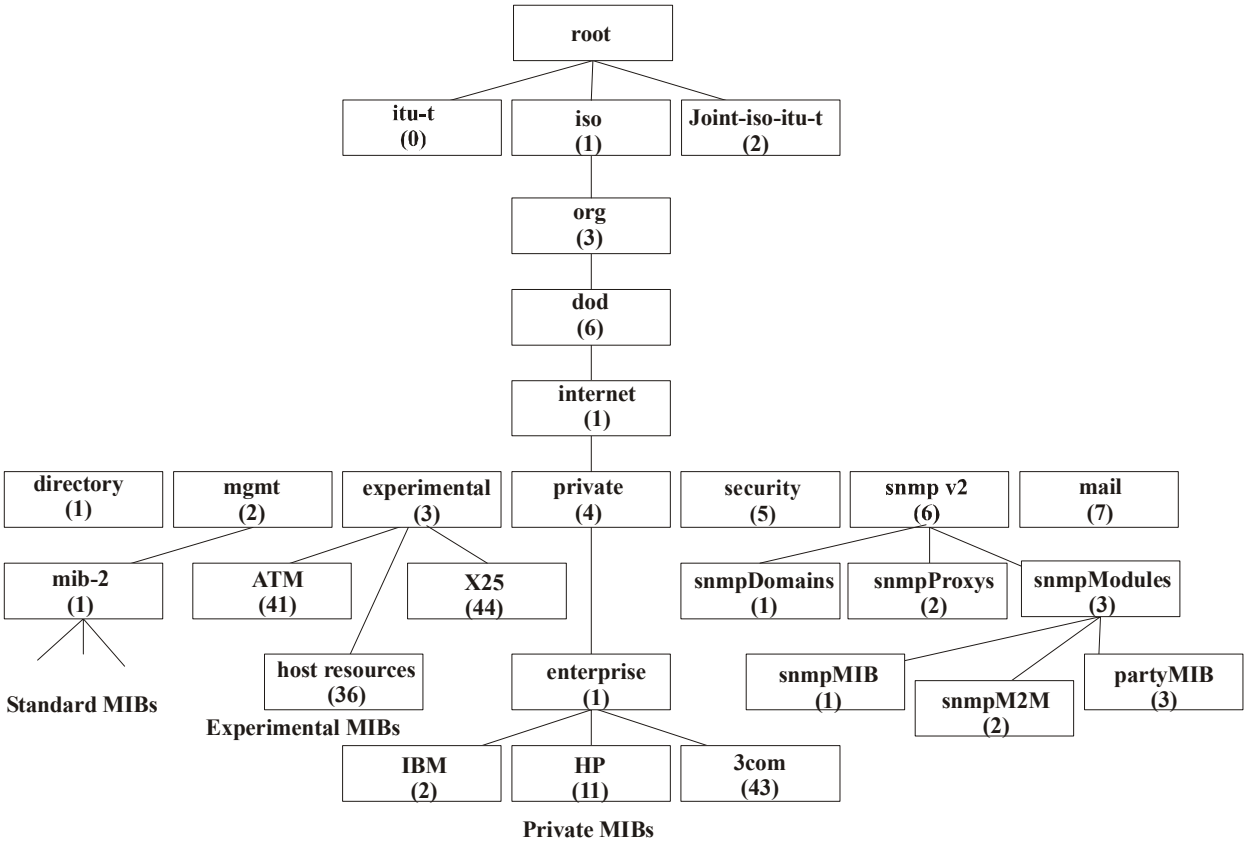
4.3 Funksjonar

Det finns ikkje strandardiserte funksjonar for feilhandtering, analyse av yting, bruksregistrering osv, slik ein har for TMN. Informasjonen og sjølve protokollen, slik dette vert omtalt i dei neste avsnitta, vil likevel kunna gi eit bilete av kva type funksjonalitet som er mogleg å byggja inn i drift- og styringsapplikasjonar basert på SNMP.

4.4 Informasjon

4.4.1 Registration tree

Styrte objekt er arrangert i den hierarkisk trestruktur, *Registration tree* som kort vart presentert i avsnitt 2.4.10.1. IETF er ansvarleg for nummerering av alle nodar under Internettnoden som har namnet *iso.org.dod.internet* og adressa *1.3.6.1* (*object identifier, OID*). Figur 4.4 syner deler av toppen av denne strukturen.



Figur 4.4 Registration tree (1)

Ein finn den standardiserte drift- og styringsinformasjonen under *mgmt*-noden 1.3.6.1.2.1. Under *experimental*-noden finns hovudsakleg teknologispesifikk drift- og styringsinformasjon, til dømes MIBs for ATM, FR og SONET. Desse basane er kandidatar for utvidingar til eksisterande *Internet Standard MIB*. I 1997 var 70 MIBs av denne typen registrert (1). I *Enterprise*-subtreet finns den leverandørspesifikke informasjonen. Alt i 1997 var det 3000 leverandørspsesifikke subtre (1). Mange av desse basane har fleire hundre objekt/variablar.

4.4.2 SNMP objekttypar

Styrte objekt av same type vert definert som ein objekttype. Objekttypane i MIB er definert ved hjelp av ASN.1, men berre eit subsett av datatypane i ASN.1 vert nytta. Ein unngår komplekse datatypar og strukturar for å leggja mest mogleg til rette for enkle implementasjonar. Informasjonsmodellen er følgjeleg svært enkel og er ikkje objektorientert. MIB kan berre lagra enkle datatypar: skalarar og todimensjonale tabellar av skalarar. Dette står i sterk kontrast til den objektorienterte og komplekse informasjonsmodellen til ISO/ITU-T.

For å identifisera objekttypane vert *Object Identifier* datatype nytta. Denne datatypen er ein sekvens av heiltal som traverserer registreringstreet, sjå figur 4.4. Dette gir ein allmenn måte å identifisera eit objekt uavhengig av semantikken til dette objektet.

Syntaksen for ein objekttype er gitt ved datatypen. Utanom datatypen vil objekttypedefinisjonen spesifisera kva som er lovleg verdisett, status, samt nokre andre aspekt ved objekttypen.

Objekt er instansar av ein objekttype og er knytt til ein verdi. Ved hjelp av agenten kan stasjonen utføra funksjonane sine ved å motta eller ved å modifisera verdien til objektet. Kvart objekt representerer *eitt aspekt* ved nettverkselementet. Eit SNMP-objekt er dermed eigentleg ein variabel og kan i prinsippet sidestillast med ein attributt for eit OSI-objekt.

Objekttypane kan spesifiserast slik at ein objektinstans representerer aggregert informasjon, til dømes i form av ei liste eller ein tabell. Protokollen må spesifisera om han støttar tilgang til aggregerte objekt og må sørgja for mekanismar som gir tilgang til dei underliggjande ikkje-aggregerte objekta. Protokollen må vidare spesifisera kva for instans som skal returnerast dersom det finns meir enn *ein* instans av objekttypen.

4.4.3 SNMPv1

4.4.3.1 Datatypar

Ein skil mellom tre grupper av lovlege datatypar (29):

- *Primitive Types*. Følgjande ASN.1-typar er lovlege:
  - *Integer* som representerer heiltal i området -2^{31} til $2^{31}-1$.
  - *Octet String* som representerer tilfeldige binære eller tekstlege data. Nokre implementasjoner avgrensar storleiken til 255 oktettar.
  - *Object Identifier* som representerer administrativt tilordna namn. Datatypen er ein sekvens av heiltal, kvart heiltal kan kallast ein subidentifikator. Einkvar instans av objekttypen kan ha opp til 128 subidentifikatorar. En subidentifikator må ikkje overskrida verdien $2^{32}-1$.
  - *Null*

  Dei primitive datatypane er gjerne kalla ikkje-aggregerte typar og kan sjåast som byggjeklossar for dei samansette datatypane.
- *Constructor Types*. Følgende ASN.1-typer er lovlege
  - *Sequence* som representerer en sekvens eller ei liste av skalare objekt av primitiv datatype. Sekvensen kan til dømes vera ei rekkje i ein todimensjonal tabell.
  - *Sequence of* som representerer ein tabell med rekkjer av skalare ”kolonneobjekt”.

  Desse datatypane er gjerne kalla aggregerte typar.
- *Defined Types.* Dette er ASN.1 datatypar som er relevante for spesielle applikasjonar. For SNMP er følgjande typar definert:
  - *NetworkAddress* som representerer ei adresse frå ein eller fleire protokollfamiliar (førebels berre frå TCP/IP-familien).
  - *IPAdress* som representerer en 32-bits internettadresse spesifisert i IP.
  - *Counter* som representerer eit ikkje-negativt heiltal som øker til en maksimumsverdi for så å starta på nytt frå 0. Maksimumsverdi er sett til $2^{32}-1$.

- *Gauge* som representerer eit ikkje-negativt heiltal som auker eller minkar og som vert låst ved en maksimumsverdi. Maksimumverdi er satt til 2^{32}-1
- *TimeTicks* som representerer eit ikkje-negativt heiltal som syner tida som har gått sidan eit spesifisert referansetidspunkt i hundredels sekundar (modulo 2^{32}).
- *Opaque* som koder en verdi til *Octet String*. Typen vert berre nytta for å sikra bakoverkompabilitet og skal derfor ikkje brukast i definisjonen av nye objekttypar.

4.4.3.2 Objekttypar

Ein objekttype vert definert ved hjelp av ASN.1 *Object Type macro*. Følgjande klausular (*clauses*) inngår (32):

- *ObjectName* som er ein obligatorisk klausul og sett saman av *Object Identifier* som gir objekttypen eit unikt administrativt tilordna namn, samt *Object Descriptor* som inneheld eit tekstleg namn.
- *Syntax* som er ein obligatorisk klausul, og som må vera i samsvar med retningslinjene for lovlege datatyper.
- *Access* som er ein obligatorisk klausul, og som kan setjast til enten *read-only*, *read-write*, *write-only* eller *not-accessible*. Verdien uttrykkjer naudsynt minimumsnivå for å kunna administrera objekttypen og kan overstyrast av den konkrete implementasjonen.
- *Status* som er ein obligatorisk klausul, og som kan setjast til enten *mandatory*, *optional, obsolete* eller *deprecated*. Verdien uttrykkjer kor vidt objekttypen må implementerast for å kunna hevda at ei aktuell objektgruppe er støtta, jfr neste avsnitt.
- *Description* som er ein valfri klausul, og som gir ein tekstleg definisjon av semantikken til objekttypen.
- *Reference* som er en valfri klausul, og som gir ein tekstleg kryssreferanse til objekt definert i andre modular, til dømes i en MIB produsert av ein annen organisasjon.
- *IndexPart* som må vera til stades berre dersom objekttypen er ei tabellrekkje. Klausulen inneheld identifikatorinformasjon for kolonneobjekta under dette objektet.
- *DefVal* som er ein valfri klausul, og som definerer default-verdiar som ein til dømes kan nytta når ein opprettar objektet.

4.4.3.3 Objekt og objektgrupper

Ein identifiserer eit MIB-objekt ved å knyta den einskilde instansen til *Object Identifier* for objekttypen. MIB-spesifikasjonen definerer ikkje korleis kvar einskild instans skal refererast. Referanse til einskildinstansar oppnår ein ved ein protokollspesifikk mekanisme.

Objekta i MIB-II kan delast inn i følgjande ti grupper:

- *System*. Gruppa er obligatorisk. Objekta gir overordna informasjon om eit system med omsyn til hardware, software, oppetid og kva type tenester systemet tilbyr på dei ulike kommunikasjonslaga.
- *Interfaces*. Gruppa er obligatorisk. Objekta er aggregert til ein tabell over dei fysiske grensesnitta til ein entitet. Kvart grensesnitt har variablar for adresse, type, bandbreidde, konfigurasjon, status og statistikk (tellarar) for inn/ut-oktettar, inn/ut-pakkar osv.
- *Address Translation*. Gruppa er obligatorisk og inneheld ein tabell for å oversetja ei nettverksadresse, for eksempel frå ei IP-adresse til ei fysisk adresse spesifikk for subnettet. Kvar rekkje i tabellen korresponderer til eitt av dei fysiske grenesnitta til entiteten. Denne gruppa er oppretthalden i MIB-II berre for å sikra kompatibilitet med MIB-I. I framtida skal

kvar protokollgruppe definerast med sin egen adresseoversettingstabell.

Gruppene *IP*, *ICMP*, *TCP*, *UDP*, *EGP*, *SNMP* vert kalla protokollgrupper. Dei inneheld hovudsakleg tellarar for ulike innkommende/utgåande protokolldataeiningar så vel som for ulike typar feilsituasjonar. I tillegg til slik statistikkinformasjon, kan gruppene òg innehalda tabellar med opplysningar om ruting, samband og nabonodar:

- *IP*. Gruppa er obligatorisk. Objekta inneheld infomasjon om datagramtrafikken til og frå ein entitet (vertsmaskin eller gateway) med omsyn til tal, feil, fragmentering osv. Gruppa inneheld dessuten tre tabellar:
  - *IP Address Table* som inneheld informasjon om alle IP-adresser som er tildelt ein entitet. Tabellen har ei rekkje pr adresse. Kvar adresse er knytt opp til eit fysisk grensesnitt.
  - *IP Address Translation Table* som i all hovudsak inneheld same informasjonen som tabellen i *Address Translation*-gruppa.
  - *IP Routing Table* som inneheld informasjon om alle ruter (primær og alternative) som er kjent for entiteten.
- *ICMP*. Gruppa er obligatorisk. Objekta er hovudsakleg tellarar som vert nytta til statistikk over ulike inn- og utgående *Internet Control Message Protocol* (ICMP)-meldinger.
- *TCP*. Gruppa er obligatorisk for alle system som implementerer TCP. Objekta inneheld informasjon om mellom anna kor mange TCP-tilknytingar entiteten støttar, time-out for retranismisjon. Her er tellarar for sende og mottekne segment samt ein tabell over tilknytingsspesifikk informasjon. Objekta som representerer informasjon om spesifikke TCP-tilknytingar er transiente, og informasjonen varer berre så lenge oppkoplinga varer.
- *UDP*. Gruppa er obligatorisk for alle system som implementerer protokollen. Objekta er hovudsakleg tellarar for datagramtrafikken til og frå entiteten.
- *EGP*. Gruppa er obligatorisk for alle system som implementerer *Exterior Gateway Protocol* (EGP). Objekta er hovudsakleg tellarar. Her finns òg ein rutingtabell samt informasjon om EGP nabonoder.
- *SNMP*. Gruppa er obligatorisk for alle system som støtter ein SNMP protokollentitet. Objekta er hovudsakleg tellarar for mottekne og sende SNMP-meldingar, samt for ulike feilsituasjonar.
- *Transmission*. Gruppa er obligatorisk for alle system som har grensesnitt mot eit transmisjonsmedium. Gruppa er nært knytt til *Interfaces*-gruppa.

MIB-II inneheld berre objekt som går for å vera essensielle, og difor er alle objekta i ei gruppe obligatoriske innan gruppa.

I tillegg til MIB-II finns nokre andre MIBs som har status som *Internet standard*. Av desse kan særleg *Ethernet Interface MIB* òg kalla *EtherLike MIB*, nemnast. Basen definerer objekt som representerer eigenskapane til eit ethernettliknande transmisjonsmedium i større detalj enn *interfaces*-gruppa i MIB-II gjer. Det finns vidare MIB-spesifikasjonar for RMON. Desse vert gjennomgått i eit seinare avsnitt.

4.4.3.4 Operasjonar på objekta

For at stasjonen skal kunna styra agenten, må heile eller deler av MIB-II-gruppene implementerast i agenten. Operasjonane på objekta er ulike variantar av *get*, *set* og *trap*-operasjonar, dvs hovudsakleg forespørslar om å setja eller henta ein parameterverdi. Istaden for

å nytta standardiserte ”kommandoar”, vert ønska aksjon iverksett ved å setja relevante parameterverdiar direkte. Kva operasjonar som er tilletne på eit objekt, går indirekte fram av *Access*-klausulen i objektdefinisjonen, jfr avsnitt 4.4.3.2. Merk at dette tilgangsnivået uttrykkjer det som gir meining for *protokollen*, og at det difor er uavhengig av administrative reglar. Dersom ein ynskjer å overstyra tilgangsnivået som er spesifisert i MIB, må ein implementera objektet på ein særskilt måte når ein definerer agenten.

4.4.4 SNMPv2

Det er ingen grunnleggjande skilnad på SNMPv1 og SNMPv2, korkje med omsyn til informasjonsstruktur eller objekttypedefinisjon. SMIv2-dokumenta er dei einaste av SNMPv2-dokumenta som har status *Internet Standard*. I motsetning til SMIv1, spesifiserer SMIv2 (44) òg korleis informasjonsmodular og notifikasjonar (traps) skal definerast. MIB for SNMPv2 (37) inneheld sjølve definisjonane.

SMIv2 inneheld tillegg på følgjande område:

- *Objekttypedefinisjonen* er i prinsippet den samme som i SNMPv1, men det vert innført nokre nye datatypar, nye klausular i objektdefinisjonen samt nokre endringar i verdisettet i eksisterande klausular.
- *Konseptuelle tabellar*. Som tidlegare handterer ein berre skalare objekt. Kompleks informasjon kan som før representerast som konseptuelle tabellar. Ein utvidar no funksjonaliteten rundt desse tabellane ved å gjera det mogleg å oppretta og sletta heile tabellrekkjer. Ein betrar vidare tilgangen til tabellrekkjene ved hjelp av nye indekseringsmekanismar.
- *Notifikasjonar* vert definert ved hjelp av *Notification Type*-makroar. Ein definerer informasjonsinnhaldet som skal sendast i ein PDU. Samanlikna med SNMPv1, der hendingene blir definert av protokollen, gir dette ein meir fleksibel spesifikasjon av SNMP-traps, jfr kap 1.5.
- *Informasjonsmodular* som spesifiserer ei gruppe av relaterte definisjonar. ASN-1 vert nytta til å definera modulane. Det er tre typer informasjonsmodular:
  - MIB-modular som inneheld definisjonar av relaterte styrte objekt. I desse modulane nyttar ein *Object Type*- og *Notification Type*-makroar.
  - Retningslinjer for å sikra samsvar mellom ulike MIB-moduler. I desse modulane nyttar ein *Object Group* - og *Module Compliance*-makroar.
  - Retningslinjer for implementasjon av agenter. I desse modulane nyttar ein *Agent Capabilities*-makroar.

Sjølv om SNMPv2 ikkje har ”teke av”, er SMIv2 derimot nytta for dei aller fleste nyare MIBs. Ein kan sjå SMIv2 som eit supersett for SMIv1. Det finns mekanismar som omset SMIv2-baserte MIBs til bruk for SNMPv1.

4.4.4.1 Datatypar

Det vert innført følgjande nye lovlege datatypar (44):

- *Integer32* (*Primitive Type*) som representerer heltall i området -2^{31} til $2^{31}-1$. Datatypen kan ikkje som *Integer*-typen nyttast til å representera opplisting av heiltalsinformasjon (*enumeration*).
- *Bits* (C*onstructorType*) som representerer ei opplisting av namngitte bitar. Nokre

implementasjonar avgrensar storleiken til 128 biter.
- *Counter32* (*Defined Type*) som representerer eit ikkje-negativt heiltal som aukar monotont til ein maksimumsverdi for så å starta frå 0. Maksimumsverdi er satt til 2^{32}-1.
- *Gauge32* (*Defined Type*) som representerer eit ikkje-negativt heiltal som aukar eller minkar og som vert låst ved ein maksimumsverdi. Maksimumverdi er sett til 2^{32}-1.
- *Counter64* (*Defined Type*) som representerer eit ikkje-negativt heiltal som auker monotont til ein maksimumsverdi for så å starta frå 0. Maksimumsverdi er sett til 2^{64}-1.
- *Unsigned32* (*Defined Type*) som representerer heiltall i området 0 til 2^{32}-1.

For fleire av desse nye datatypane er det berre namnet som er endra frå SMIv1.

4.4.4.2 Tekstkonvensjonar

SMIv2 introduserer tekstkonvensjonar (*terxtual conventions*). Desse kan sjåast som ei forfining av dei enkle datatypane og formålet er å kunna definera objekt på ein meir standardisert og abstrakt måte.

Tabell 4.1 gir ein oversikt over definerte tekstkonvensjonar (45).

Tekstkonvensjon	**Kommentar**	**Datatype**
Display String	Representerer tekstinformasjon (ASCII).	*Octet String (255 karakterar)*
PhysAdress	Representerer ei mediia- eller fysisk adresse.	*Octet String*
MacAdress	Representerer ei 802 Medium Access Control (MAC-adr).	*Octet String (6 bytes)*
TruthValue	Representerer ein boolsk verdi.	*Integer*
TestAndIncr	Representerer ein verdi som vert nytta for å hindra at to stasjonar prøver å modifisera same objektet samstundes.	*Integer*
AutonomousType	Representerer identifikasjonsinformasjon.	*Object Identifier*
VariablePointer	Peikar til eit spesifikt objekt.	*Object Identifier*
RowPointer	Peikar til ei konsptuell tabellrekkje	
RowStatus	Representerer ein standardisert kode for status til ei konspetuell tabellrekkje. Verdien vert nytta ved oppretting, aktivisering og sletting av rekkjer.	*Integer*
TimeStamp	Representerer ein verdi som syner kor lengje ein SNMP-agent har gått sidan ei spesifikk hending oppstod.	*TimeTicks*
TimeInterval	Representerer ein tidsperiode i hundredelssekundar.	*Integer*
DateAndTime	Representerer dato og tid; tidssone evt i bytes 9-11	*Octet String (8 | 11 bytes)*
StorageType	Representerer ein standardisert kode for korleis ei tabellrekkje skal lagrast.	*Integer*
TDomain	Representerer type transportteneste, til dømes ”SNMP over UDP”	*Object Identifier*
TAdress	Representerer adresse for transportteneste.	*Octet String (255 karakterar)*

Tabell 4.1 Oversikt over tekstkonvensjonar

4.4.4.3 Objekttypar

Det vert innført følgjande endringar i klausulane (44):
- *Max-Access* tilsvarer *Access*-klausulen i SMIv1. Verdisettet er endra til *read-create, read-write*, *read-only, accessible-for-notify* eller *not-accessible*. Merk at verdien no representerer det *maksimale* tilgangsnivået for objektet. Dette gjeld uavhengig av administrativ praksis for autorisasjon.
- *Status* tilsvarer *Status*-klausulen i SMIv1. Verdisettet er endra til *current, obsolete* eller *deprecated*.

- *IndexPart* tilsvarer *IndexPart*-klausulen i SMIv1 men er meir kompleks på grunn av den utvida funksjonaliteten for handtering av tabellar.
- *Augment* er ein ny klausul og er eit alternativ til *IndexPart*. Klausulen vert mellom anna brukt ved proprietære utvidingar av definerte tabellrekkjer (nye kolonneobjekt).
- *Units* er ein ny valfri klausul og er ein tekstleg definisjon av einingar som kan assosierast med objektet.

4.4.4.4 Objekt og objektgrupper

Dei aller fleste implementasjonar støttar framleis standard MIB-II objektgrupper, sjå avsnitt 4.4.3.3. SNMPv2 inneber nokre endringar til standarden. Gruppene *EGP* og *Address Translation* går ut. (37) omhandlar endringane i gruppene for *System* og *SNMP* samt den nye gruppa *SNMP MIB Objects*. Definisjonane av dei andre gruppene er flytta til separate dokument. Endringane kan samanfattast slik:

- *System*. Gruppa får eit tillegg i form av *Object Resource Information*. Desse objekta dannar ein tabell som syner i kva grad SNMPv2-entiteten støttar ulike MIB-modular.
- *SNMP*. Gruppa vert redefinert og får i tillegg nokre nye tellarar.
- *SNMP MIB Objects*. Dette er ei ny gruppe som inneheld objekt som er relevante for styring av MIB-objekt. Gruppa har følgjande undergrupper:
  - *SNMP Trap*. Objekta som er definert her skal setja SNMPv2-entitetar i agentrolle i stand til å generera SNMPv2-Trap-PDUs.
  - *SNMP Traps*. Definerer SNMPv1-trap som MIB Objects.
  - *SNMP Set*. Objekta skal tillata ulike samarbeidande SNMPv2-entiteter i managerrolle å koordinera bruken av SNMPv2 *set*-operasjon.
- *Interfaces*. Gruppa, som no kallast *Interfaces MIB*, får nokre nye tellarar og tabellar.
- *IP*. *IP Routing Table* vert erstatta av *IP Forwarding MIB*.

4.4.4.5 Operasjonar på objekta

SMIv2 formaliserer definisjonen av kva agenten skal vera i stand til gjennom bruken av *Agent Capability*-makro, noko som kan gi store fordeler med omsyn til interoperabilitet (46). Objekt og notifikasjonar som agenten støttar, må spesifiserast ved å referera til MIB(s) og objektgruppe(r). Det må vidare spesifiserast dersom agenten skal ha redusert tilgang til eit objekt i høve til objektet sin *MaxAccess*-klausul.

4.4.4.6 Notifikasjonstypar

I tillegg til namnet, Object Identifier, inngår følgjande klausular i definisjonen (44):

- *Objects* som er en valfri klausul, og som definerer ein sekvens objekttypar som skal vera med i kvar instans av notifikasjonstypen.
- *Status* som er ein obligatorisk klausul, og som kan setjast til enten *current, obsolete* eller *deprecated*.
- *Description* som er ein obligatorisk klausul, og som gir ein tekstleg definisjon av semantikken til notifikasjonen. Klausulen bør spesielt spesifisera kva konkrete instansar av objekttypane definert i *objects*-klausulen, denne notifikasjonstypen skal innehalda.
- *Reference* som er en valgfri klausul, og som gir en tekstlig kryssreferanse til notifikasjoner definert i andre moduler, for eksempel i en MIB produsert av en annen organisasjon.

4.5 Kommunikasjon

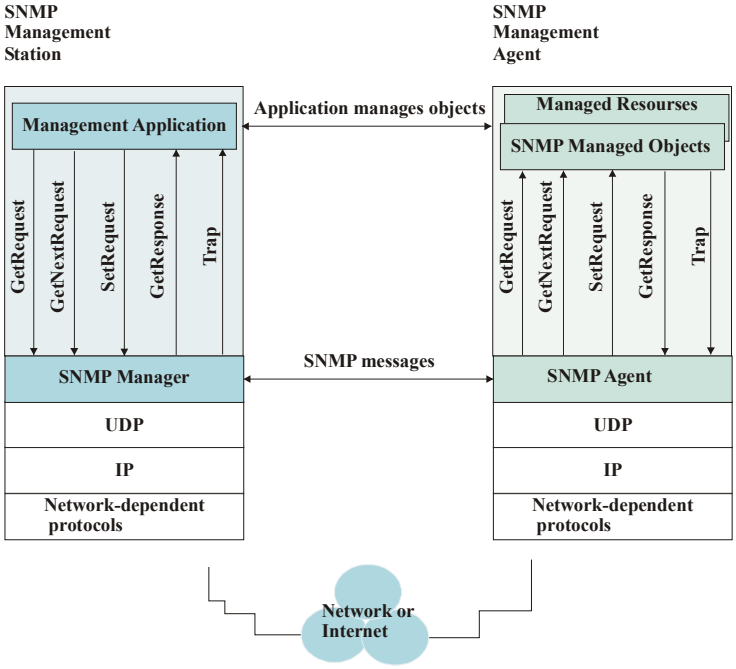
4.5.1 Meldingar

For å utveksla SNMP-meldingar krevs berre ei upåliteleg datagramteneste. Kvar melding er fullstendig og uavhengig representert av eitt einskild transportdatagram (31). Sjølv om ein tilrår å utveksla meldingane ved hjelp av UDP, er mekanismane i SNMP utforma slik at ein kan nytta ei rekkje andre transporttjenester.

Ved hjelp av meldingane kan ein inspisera og endra objekta i agenten sin MIB. Figur 4.5 syner rollen til SNMP.

Ei melding er sett saman av:

- *Version Identifier* som syner SNMP-versjonen (for SNMPv1 er dette 0).
- *SNMP community name* som er ein *Octet String* som tener som eit passord for å autentisera SNMP meldinga.
- *PDU* som spesifiserer om meldinga dreier seg om ein operasjon (get, get-next,set) eller trap.



Figur 4.5 Kommunikasjon med SNMP (3)

Meldingsutvekslinga går føre seg etter følgjande prosess:

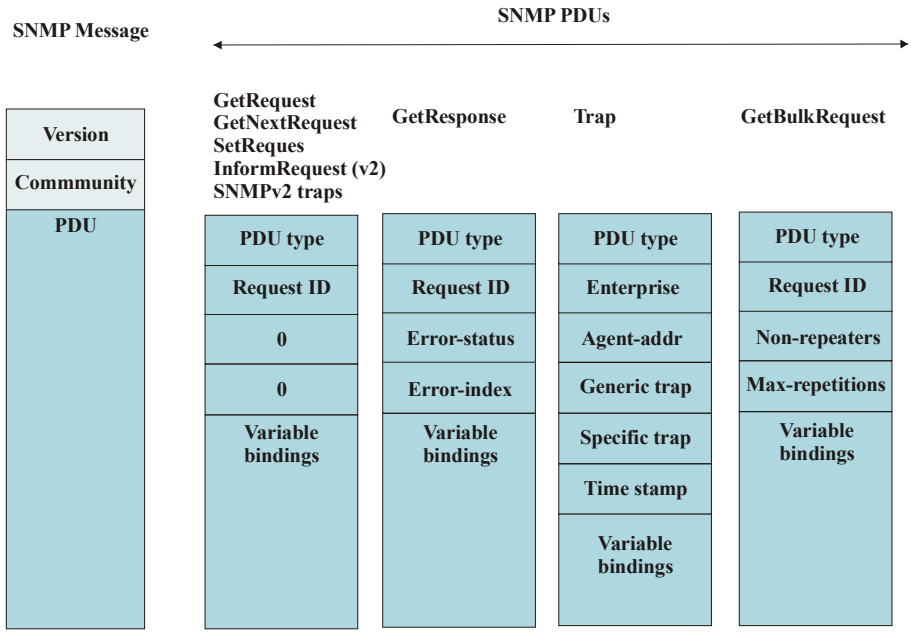
- Ein PDU vert konstruert ved hjelp av ASN.1 slik dette er spesifisert i (31).
- PDU vert så levert til ei autentiseringsteneste saman med transportadressene for kjelde og destinasjon samt namnet på SNMP fellesskapet (SNMP community name). Når UDP er nytta, vil ei transportadresse berre vera IP-adressa og aktuell UDP-port. Autentiseringstenesta utfører ønska transformasjon som kryptering eller innsetjing av autentiseringskode og returnerer resultatet. I praksis vert autentisering ikkje utført.
- Protokollentiteten konstruerer ei melding på bakgrunn av foregåande resultat.
- Meldinga vert så koda til eit nytt ASN.1-objekt og overført til transporttenesta.

Meldingsformatet som er synt i figur 4.6 vert nytta for både SNMPv1 og SNMPv2. Ein brukar gjerne omgrepet *community-based* SNMPv2 eller SNMPv2C. Dette skuldast at ein har måtta basera datasikringa i SNMPv2 på *community*-konseptet frå SNMPv1, sidan ein som tidlegare nemnt ikkje oppnådde semje om datasikring i arbeidet med versjon 2. Denne forma for datasikring vert omtalt i avsnitt 4.7.

4.5.2 SNMPv1

4.5.2.1 PDU-formatet

Figur 4.6 syner meldings- og PDU-formata.



Figur 4.6 Meldings- og PDU-formata i SNMPv1 og v2

PDUs er sett saman av:

- *Request-id* som er nytta til å skilja uteståande forespørslar frå kvarandre.
- *Error-status* som er nytta for å indikera om feil har oppstått under prosessering av forepørselen. Her finns 5 ulike statusverdiar som indikerer feiltype.
- *Error-index* som vert nytta for tilleggsinfomasjon når feil har oppstått.
- *Variable bindings* som er ei liste av variabelnamn og korresponderande verdiar.
- *Enterprise* som indikerer objektet som genererer trap.
- *Agent-addr* som er adressa til objektet som genererer trap.
- *Generic trap* som gir ein av sju ulike traptypar.
- *Specific trap* som gir ein trapkode.
- *Time stamp* som syner tida som har gått frå siste (re-)initialisering av nettelementet til generering av aktuell trap.

4.5.2.2 Get-operasjonar

Følgjande PDU-typar er obligatoriske:

- *Get Request* vert generert av ein protokollentitet på forespørsel frå ein applikasjonsentitet. Mottakende protokollentitet svarar med å senda *Get Response*. *Get Response PDU* skal levera verdien for kvart objekt nemnt i variablellista i *Get Request PDU*. Merk at dersom variabellista innehelder eit aggregert objekt, vil dette medføra at feilindikator vert sett sidan protokollen berre handterer skalare verdiar.
- *Get Next Request* vert generert av ein protokollentitet på forespørsel frå ein applikasjonsentitet og har samme form som *Get Request*. Mottakende protokollentitet svarar

med å senda *Get Response*. For kvart objekt nemnt i vaariabellista i *Get Request PDU*, skal *Get Response PDU* levera verdien til *påfølgjande* objekt i ei liste over objekt som er tilgjengelege for Get-operasjonar i det aktuelle MIB-utsnittet (MIB view). *Get Next Request* vert mellom anna nytta til å traversa tabellar i MIB. Kolonneobjekta i ei tabellrekkje må då spesifiserast og agenten vil levera verdiane i påfølgjande rekkje. Merk at ein må senda ein ny *Get Next Request* for kvar tabellrekkje en vil sjå.

- *Get Response* vert generert av ein protokollentitet ved mottak av *Get Request*, *Get Next Request* eller *Set Request*, og har samme form som *Get Request*. Mottakande protokollentitet presenterer innhaldet til sin applikasjonsentitet.

4.5.2.3 Set-operasjonar

Følgjande PDU-type er obligatorisk:

- *Set Request* vert generert av ein protokollentitet på forespørsel frå ein applikasjonsentitet og har samme form som *Get Request*. Dei ny verdiane skal tilordnast for kvart objekt nemnt i variabellista til S*et Request PDU*. Mottakande protokollentitet svarar med å senda *Get Response*.

4.5.2.4 Traps

Trap vert generert av ein protokollentitet på forespørsel frå ein applikasjonsentitet. Mottakande protokollentitet presenterer innhaldet til sin applikasjonsentitet. *Trap PDU* har eit felt kalla *generic trap* som syner kva type *trap* som vert sendt. Følgjende typar er definert:

- *Cold Start* som meddeler at sendar-protokollentiteten re-initialiserer seg sjølv slik at agentkonfigurasjonen eller implementasjonen av protokollentiteten kan verta endra.
- *Warm Start* som meddeler at sendar-protokollentiteten re-initialiserer seg sjølv slik at korkje agentkonfigurasjonen eller implementasjonen av protokollentiteten kan verta endra.
- *Link Down* som meddeler at sendar-protokollentiteten har funne ein feil på ein kommunikasjonslink som er repreresentert i agentkonfigurasjonen.
- *Link Up* som meddeler at ein kommunikasjonslink som er repreresentert i agentkonfigurasjonen er oppe att.
- *Authentification Failure* som meddeler at sendar-protokollentitet har motteke ei melding som ikkje er tilstrekkeleg autentisert.
- *EGP Neighbour Loss* som meddeler at ein EGP-nabo av sendar-protokollentiteten er nede.
- *Enterprise Spesific* som meddeler at sender-protokollentitet kjenner att ei hending som definerte spesifikt for verksemda. *Trap PDU* har òg eit felt kalla *spesific trap*. Dette feltet vil då syna kva trap det dreier seg om.

4.5.2.5 Identifikasjon av objekta i MIB

I ein SNMP-operasjon vert kvar instans av ein objekttype som er definert i MIB, identifisert ved eit unikt namn kalla *variabelnamn*. Generelt er eit namn på ein SNMP-variabel ein *Object Identifier* på forma *x.y*, der *x* er er namnet til ein ikkje-aggregert objekt-type definert i MIB, og *y* er eit fragment som identifiserer den ønska instansen (objektet).

4.5.3 SNMPv2

Sjølv om UDP framleis vert føretrekt som transportprotokoll, definerer (36) fleire alternativ SNMPv2 kan nytta. Desse er mellom anna OSI-protokollane CLNS og CONS samt *Novell Internetwork Packet Service* (IPX) og *Appletalk*.

(35) definerer gir tre ulike typar tilgang til drift- og styringsinformasjonen (35):
- *Manager-agent request-response* der ein SNMPv2-entitet i managerrolle sender ein forespørsel til ein SNMPv2-entitet i agentrolle, som i sin tur responderer på denne. Denne typen interaksjon er kjend frå SNMPv1 og vert nytta til å henta eller til å endra informasjon i det styrte elementet, der agenten er implementert.
- *Manager-manager request-repsonse* der ein SNMPv2-entitet i managerrolle sender ein forespørsel til ein annan SNMPv2-entitet i managerrolle, som i sin tur responderer på denne. Denne typen interaksjon er ny, og vert nytta for kommunikasjon mellom to SNMPv2-entitetar som beggje har managerrolle, sjå figur 4.3.
- *Agent-manager unconfirmed* der ein SNMPv2-entitet i agentrolle uoppfordra sender ei melding, trap, til ein SNMPv2-entitet i managerrolle. Denne typen er kjend frå SNMPv1 og vert nytta for å meddela hendingar som har medført informasjonen vedrørande det styrte elementet er endra.

4.5.3.1 Operasjonar

SNMPv2 kjem med følgjande endringar og tillegg:
- *Get Request*, *Get Next Request* og *Response* (tidlegare *GetResponse*) har same format og semantikk som i SNMPv1. Ein del nye kodar er definert for feilstatusfeltet i *Response PDU*. Her er vidare nokre endringar vedrørande handtering av varibellistene til *Get Request PDU* og *Get Next Request PDU*.
- *Get Bulk Request* vert generert og sendt på forespørsel frå ein SNMPv2 applikasjonsentitet. Dette er ein ny operasjon, og formålet er å kunna senda større mengder med data, til dømes store tabellar. Mottakande entitet svarar med å senda *Response*. Variabellista i *Get Bulk Request PDU* vert spesifisert og handtert noko annleis enn tilsvarande lister i dei før nemnde operasjonane. Formatet for *Get Bulk Request PDU* vart synt i figur 4.6. Her er to nye felt som ikkje fanns i SNMPv1-formata:
  - *Non-repeaters* som spesifiserer talet på variablar i variabellista der det skal returnerast ein einskild leksikografisk påfølgjande variabelverdi.
  - *Max-repetitions* som spesifiserer talet på leksikografisk påfølgjande variabelverdiar som skal returnerast for kvar av dei resterande variablane i variabellista.
- *Set Request* har same format og semantikk som i SNMPv1. Einaste skilnaden er i måten *Response* vert handtert. Mottakande entitet utfører ein meir omfattande valideringsprosess enn tidlegare, jfr dei mange nye feilstatusane i *Response PDU*.
- *Inform Request* vert generert og sendt av ein SNMPv2-applikasjonsentitet med managerrolle til ein annan SNMPv2-entitet med managerrolle. Dette er ein ny operasjon, og formålet er å meddela informasjon mellom entietetar med managerrolle. PDU-formatet er det same som *Get/Set Request* og *Get Next Request*-operasjonane har. Mottakande entitet svarar med å senda *Response*.
- *SNMPv2 trap* vert generert og sendt av ein SNMPv2-entitet med agentrolle til ein SNMPv2-entitet med managerrolle og har same funksjon som tidlegare SNMP *trap*. Ein går no bort frå det særeigne formatet for *Trap PDU* og nyttar istaden same PDU-format som *Get/Set Request* og *Get Next Request*-operasjonane har. Informasjonen ein tidlegare fann i spesialfelt, finn ein no i variabellista.

SNMPv2 er ikkje utan vidare kompatibel med SNMPv1. Skal ein handtera beggje

protokollversjonar, må ein:
- Utføra mindre endringar i SNMPv1 MIB for å få denne i samsvar med SNMPv2 SMI, sjå avsnitt 4.4.4.
- Bruka ein proxy-agent som kommuniserer med SNMPv2-entitetar i managerrolle og med SNMPv1-entitetar i agentrolle.

4.6 Remote Network Monitoring (RMON)

Remote Monitoring (RMON) er utvikla for å preprosessera informasjon i agentane. Til ein viss grad kan RMON samanliknast med nokre av OSI systemfunksjonar, som vart gjennomgått i avsnitt 2.3. RMON er ikkje knytt særskilt til korkje SNMPv1 eller SNMPv2. RMON-spesifikasjonen er det viktigaste tillegget til basis-spesifikasjonane SMI, MIB og SNMP. Standarden definerer ein RMON-MIB som supplerer *Internet Standard* MIB.

Ein vanleg standard-MIB gir berre informasjon om det einskilde nettelementet. RMON-MIB gir informasjon om nettet. RMON-MIB spesifiserer styrte objekt i ei (fjerntliggjande) monitoreringsinnretning og definerer dessutan sjølve resultata av statistiske kalkulasjonar som styrte objekt. Informasjonsinnhaldet i RMON-MIB er relatert til trafikkstatistikk, alarmar og filter.

Eit system som implementerer ein RMON-MIB vert kalla ein RMON-*probe*. I praksis vil skilnaden frå ein vanleg SNMP-agent berre vera tilgangen til RMON-informasjon og evna til å utføra RMON-relaterte funksjonar på denne informasjonen.

RMON kan tolkast som ei utviding i retning drift- og styringsfunksjonane i OSI-modellen. Trass i at RMON fører til ei stor funksjonell forbetring, medfører det ikkje endringar i protokollen.

RMON-MIB finns i to versjonar:
- RMON 1 (48) definerer objekt for overvaking og analyse av LAN, dvs for OSI lag 1 og 2. Spesifikasjonen har status som *Internet Standard*.
- RMON 2 (38) utvidar dette til OSI lag 3-7. Spesifikasjonen har status som *Proposed* S*tandard*

4.6.1 RMON 1

4.6.1.1 Tekstkonvensjonar

(48) inneheld to nye tekstkonvensjonar:

Tekstkonvensjon	Kommentar	Datatype
OwnerString	Representerer eit administrativt tilordna namn på eigaren av ein ressurs.	*Octet String (128 karakterar)*
EntryStatus	Representerer status for ein tabellinngang.	*Integer*

Tabell 4.2 RMON 1 tekstkonvensjonar

4.6.1.2 Objekt og objektgrupper

Objekta i RMON 1-MIB kan delast inn i følgjande grupper som alle er valfrie:

- *Statistics*. Objekta, som hovudsakleg er pakketeljarar, er ordna i ein tabell som inneheld statistisk informasjon basert på data frå kvart grensesnitt som vert monitorert av denne innretninga. Nokre av desse objekta finns òg i *Interface*-gruppa i MIB-II. I MIB-II er objekta relatert til kvart einskild nettelement, medan trafikklast og feilratar *for heile LAN-segmentet* er i fokus for RMON-MIB.
- *History Control*. Formålet med denne gruppa er å spesifisera og styra periodiske stikkprøvane av grensesnitt (subnett) i spesifiserte tidsperiodar.
- *Ethernet History*. Objekta inneheld informasjon frå periodiske stikkprøvar i nettet. Denne informasjonen skal gi grunnlag for trendanalysar. Objekta, som hovudsakleg er pakketeljarar, er ordna i ein tabell.
- *Host*. Objekta er ordna i tabellar som syner vertsmaskiner i nettet. Ei liste over MAC-adresser vert bygt opp ved å observera kjelde- og destinasjonsadressene for pakkane. Det finns statisisk informasjon for kvar adresse som er oppdaga i nettet. Ein eigen tabell spesifiserer grensesnitta som skal monitorerast for adresser.
- *Host TopN*. Gruppa krev at *Host*-gruppa er implementert. Objekta inneheld statistikk for valfritt mange (*N*) vertsmaskiner som toppar ei liste på grunnlag av ein gitt parameter, til dømes dei ti maskinene som sende mest data i løpet av eit gitt tidsrom.
- *Matrix*. Objekta er ordna i tabellar som lagrar statistikk basert på pakketellarar for trafikken mellom par av MAC-adresser.
- *Filter*. Objekta inneheld informasjon som gjer det mogleg å definera pakkefilter i form av logiske uttrykk. Gitte filter kan setjast saman. Ein pakkestraumar som passerer gjennom eit filter, vert referert til som ein kanal.
- *Packet Capture*. Gruppa krev at *Filter*-gruppa er implementert. Formålet med gruppa er å kunna setja opp ein bufringsplan for pakkar som vert fanga opp frå ulike kanalar, sjå *Filter*-gruppa. Objekta er ordna i to tabellar. Den eine inneheld detaljar vedrørande bufringsfunksjonen, den andre lagrar pakkedataene samt data om pakkane.
- *Event*. Gruppa støttar definisjonen av *events*. Objekta inneheld informasjon om parametrane som skal trigga *events*. Gruppa inneheld òg objekt som er ordna i ein logg-tabell der *events* vert lista. Vilkåra for at events skal genererast er spesifisert i objekta i *Alarm*-gruppa.
- *Alarm*. Gruppa krev at *Event*-gruppa er implementert. Objekta innheld resultat frå stikkprøvar samt ei rekkje terskelverdiar. Formålet er å samanlikna stikkprøveverdiane mot terskelverdiane for å generera alarmar (*events*) dersom ein terskelverdi vert overskriden. Alarmane vert sendt som *traps*.

To traps er dessutan definert: *RisingAlarm* og *FallingAlarm*.

4.6.2 RMON 2

RMON 2 MIB er utviding av RMON 1 MIB. Nokre tekstkonvensjonar og ei rekkje nye objektgrupper vert lagt til.

4.6.2.1 Tekstkonvensjonar

(52) inneheld fire nye tekstkonvensjonar:

Tekstkonvensjon	Kommentar	Datatype
ZeroBasedCounter32	Representerer ein tellar for hendingar. Tellaren vert sett til *zero* (0) ved oppretting og går tilbake til *zero* (0) når verdien 2^{32} er nådd.	*Gauge32*
LastCreateTime	Representerer tida då ei tabellrekkje sist vart oppretta.	*TimeStamp*
TimeFilter	Representerer eit tidspunkt. Objekt med denne syntaksen kan nyttast slik at kan lesa berre dei tabellrekkene som er endra etter dette tidspunktet.	*TimeTicks*
DataSource	Representerer kjelda for data som skal analyserast av ein funksjon.	*ObjectIdentifier*

Tabell 4.3 RMON 2 tekstkonvensjonar

4.6.2.2 Objekt og objektgrupper

Objekta i RMON 2-MIB kan delast inn i følgjande grupper:

- *Protocol Directory*. Objekta er ordna i ein tabell over protokollane som kan monitorerast av denne entiteten. Ein kan oppretta, sletta og konfigurera rekkene i denne tabellen. Gruppa dekkar MAC- nettverks- og høgare lags protokollar.
- *Protocol Distribution*. Objekta er ordna i ein tabell og summerer kor mange oktettar og pakkar som har vore sendt over kvar av protokollane. tabellen.
- *Adress Map*. Formålet med desse objekta er å tilordna kvar nettverksadresse ei spesifikk MAC-adresse og syna grensesnittet der desse adressene sist vart observert.
- *Network Layer Host*. Objekta er ordna i tabellar og summerer trafikken til og frå kvar nettverksadresse.
- *Application Layer Host*. Objekta er ordna i ein tabell og summerer trafikken til og frå kvar nettverksadresse for kvar applikasjonsprotokoll. Denne gruppe er avhengig av *Network Layer Host*-gruppa.
- *Network Layer Matrix*. Objekta er ordna i tabellar og summerer trafikken mellom kvart par av nettverksadresser.
- *Application Layer Matrix*. Objekta er ordna i tabellar og summerer trafikken mellom kvart par av nettverksadresser for kvar applikasjonsprotokoll. Denne gruppe er avhengig av *Network Layer Matrix*.
- *User History Collection*. Objekta er ordna i ein hierakisk tre-nivå-struktur av tabellar. Spesifikke statistikkar og variablar vert undersøkt og deretter loggført basert på brukardefinerte parametrar. Formålet med gruppa er at brukaren skal kunna konfigurera historiske studiar av ein kvar teljar i systemet.
- *Probe Configuration*. Objekta er ordna i ein tabell og gjer det mogleg å konfigurera parametrane i monitoreringsagenten.

I tillegg til dei nye objektgruppene inneheld RMON-2-spesifikasjonen òg nokre nye objekt i eksisterande objektgrupper i RMON-1-MIB.

4.7 Datasikring

Sikringsmekanismen som er nytta i SNMPv1 og v2 er basert på ei triviell autentiseringsalgoritme. Som nemnt tidlegare vert kvar PDU pakka inn i ei SNMP-melding

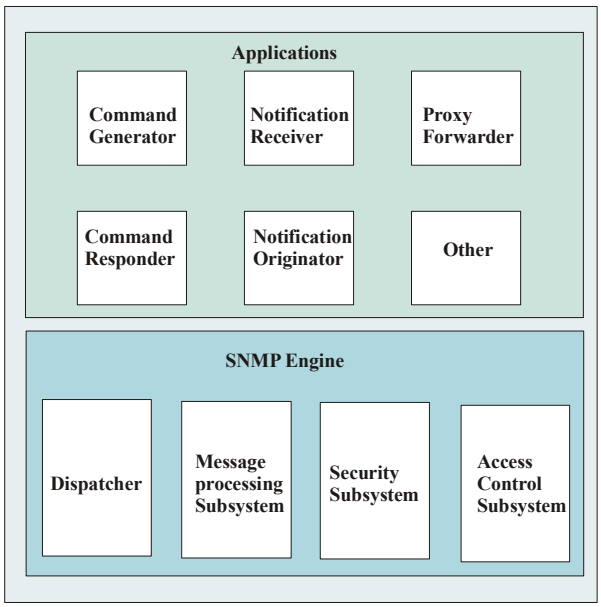
saman med eit versjonsnummer og eit fellesskapssnamn (*community name*). Dette namnet fungerer som eit passord for MIB-tilgang sidan agenten berre prosesserer SNMP-meldingar der fellesskapssnamnet høver til ein streng konfigurert i agenten. Ein svært openlys veikskap er at passordet er felles mange brukarar. Dette er einaste form for datasikring i versjon 1, og dette er ei viktig årsak til at svært mange ikkje tillet *Set*-operasjonar i SNMP-systemet sitt.

Det vil vera lett for ein uautorisert å avlytta felleskapsnamnet. Ein inntrengjar kan dermed konfigurera agenten og dermed styra denne. Prosedyren manglar vidare mekanismar som kan handtera forsøk på å forfalska meldingar, å repetera meldingar og å avlytta meldingar. SNMP-informasjonen er i praksis berre verna av sikringsmekanismane IP tilbyr, til dømes IPsec.

Informasjonssikring i SNMPv3 er svært mykje betre og vert gjennomgått i neste avsnitt.

4.8 SNMPv3

4.8.1 Arkitektur



Figur 4.7 Blokkdiagram for SNMP-entitet (42)

Arkitekturen er skildra i (42), og er sett saman av ei samling samarbeidande SNMP-entitetar. Kvar entitet implementerer ein del av kapabiliteten og kan fungera som manager, agent eller som ein kombinasjon av beggje deler. Figur 4.7 syner dei ulike modulane ein entitet er sett saman av:

Kvar entitet inneheld ei einskild *SNMP Engine* som sender, mottek, autentiserer, krypterer/dekrypterer meldingar og styrer tilgangen til styrte objekt. Desse funksjonane vert levert som tenester til *SNMP Applications*. Desse applikasjonane er definert som ei samling modular. Kva modular som vert implementert i ein SNMP-entitet, avheng av kva rolle entiteten har (manager, agent).

4.8.1.1 SNMP Engine

Her finns følgjande modular:

- *Dispatcher* som har ansvar for å godkjenna PDUs frå applikasjonane for transmisjon over nettet, vidaresenda utgåande PDUs til *Message Processing Subsystem*, vidaresenda innkomande meldingar til *Message Processing Subsystem* for å dra ut PDU, samt senda og motta SNMP-meldingar over nettet.
- *Message Processing Subsystem* som har ansvar for å førebu meldingar som skal sendast samt å ekstrahera data frå innkomande meldingar.
- *Security Subsystem* som har ansvar for tryggingstenester, som til dømes autentisering av meldingar.

- *Access Control Subsystem* som tilbyr autorisasjonstenester som applikasjonane nyttar for å kontrollerea tilgangsrettane til informasjonen. Tilgangskontroll kan verta påkalla ved alle typar operasjonar mot informasjonen.

Tenestene mellom modulane i SNMP Engine er definert i form av primitiv og parametrar. Eit primitiv spesifiserer funksjonen som skal utførast, medan tilhøyrande parametrar vert nytta til å overføra data og kontrollinformasjon.

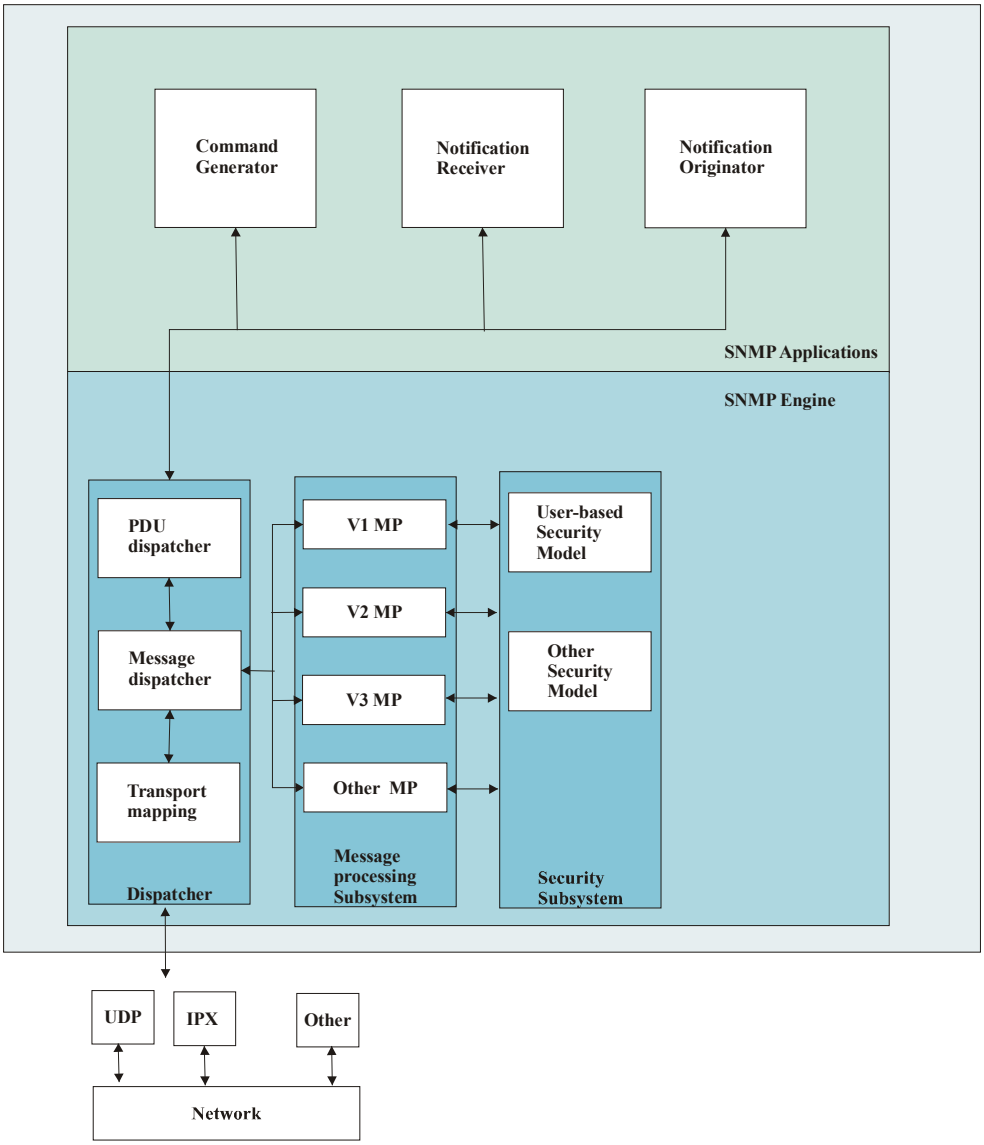
4.8.1.2 SNMP Applications

Her finns følgjande modular:

- *Command Generator* som initierer *SNMP Get-*, *GetNext-*, *GetBulk-* og/eller SetRequest PDUs og prosesserer responsen på ein forespørsel som er generert.
- *Command Responder* som mottek *SNMP Get-*, *GetNext-*, *GetBulk-* og/eller SetRequest PDUs, utfører naudsynte protokolloperasjonar, brukar *Access Control Subsystem* og genererer ei responsmelding til den som har sendt forespørselen.
- *Notification Originator* som monitorerer eit system for spesielle hendingar eller vilkår og genererer *Traps* og/eller *Inform*-meldingar basert på desse hendingane eller vilkåra. Applikasjonen må ha ein mekanisme for å avgjera kvar meldingane skal sendast samt kva SNMP-versjon og datasirkingsparametrar som skal nyttast for meldinga.
- *Notification Receiver* som ”lyttar” etter notifikasjonar og genererer ein respons dersom det dreier seg om ei *Inform*-melding.
- *Proxy Forwarder* som vidaresender SNMP-meldingar. Det er valfritt å implementera denne applikasjonen.

4.8.1.3 Manager

Figur 4.8 syner eit blokkdiagram for manager. For utgåande meldingar vil *Dispatcher* motta og akseptera PDUs frå applikasjonane, og for kvar PDU avgjera kva for SNMP-versjon som skal nyttast for å prosessera meldinga. Deretter vil PDU verta sendt til aktuell modul i *Message Processing Subsystem*. Denne modulen vil i sin tur tilføra PDU eit meldingshovud. Avhengig av parameterverdiar vil meldinga verta oversendt *Security Subsystem* for kryptering av PDU og/eller innsetjing av autentiseringskode i meldingshovudet. Meldinga vert så returnert via *Message Processing Subsystem* til *Dispatcher* som mappar meldinga til transportlaget.



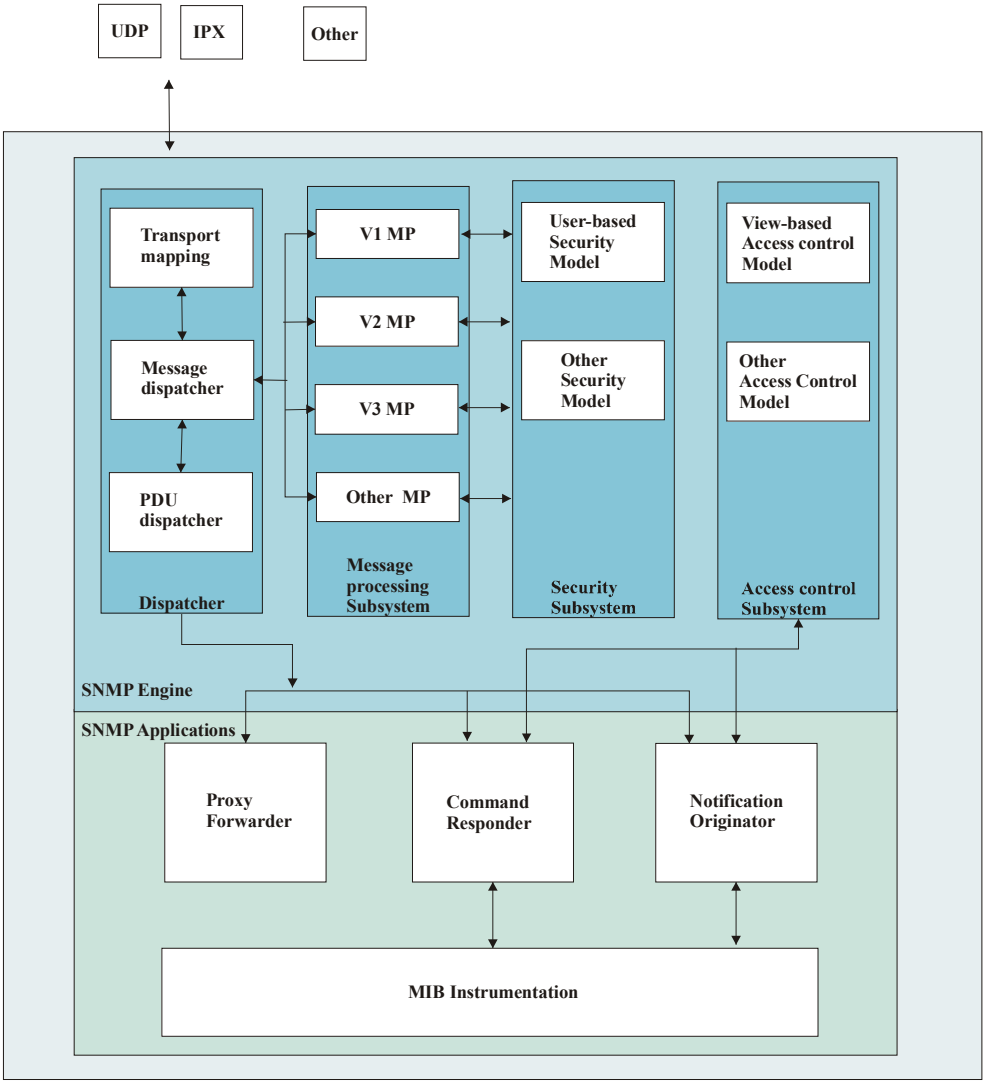
Figur 4.8 SNMPv3 manager (42)

For innkomande meldingar vil *Dispatcher* motta og akseptera meldingar frå transportlaget og ruta kvar melding til aktuell modul i *Message Processing Subsystem* som i sin tur prosesserer meldingshovudet. Avhengig av parameterverdiar vil meldinga verta oversendt *Security Subsystem* for dekryptering av PDU og/eller kontroll av autentiseringskode. PDU vert så

returnert via *Message Processing Subsystem* til *Dispatcher* som vidaresender PDU til korrekt applikasjon.

4.8.1.4 Agent

Figur 4.9 syner eit blokkdiagram for agent. Ein ser at applikasjonane avvik litt frå manager-konfigurasjonen medan SNMP Engine inneheld dei same modulane med tillegg av *Access Control Subsystem*. Dette subsystemet kontrollerer tilgangen til dei styrte objekta i MIB.



Figur 4.9 SNMPv3 agent (42)

4.8.1.5 MIBs

Det vert definert tre MIBs som skal støtta SNMPv3-applikasjonane:

- *Management Target MIB* som inneheld objekt for å kunna definera mottakarar (mål) for drift-og styringsmeldingar. Her er to typar informasjon:
  - Destinasjonsinformasjon som melloma anna omfattar transportadresser.
  - Meldingsparametrar som mellom anna gjeld prosesseringsmodell, datasikringsmodell, sikringsnivå.
- *Notification MIB* som inneheld objekt for å kunna fjernkonfigura parametrane SNMP-

entiteten nyttar til å generera notifikasjonar.
- *Proxy MIB* som inneheld objekt for å kunna fjernkonfigura parametrane SNMP-entiteten nyttar til å vidaresenda operasjonar (når entiteten er ein proxy-agent).

4.8.2 User-Based Security Model (USM)

(39) definerer fire overordna sikringstenester:
- *Dataintegritet* som vil seia at data ikkje er øydelagde eller endra på uautorisert vis.
- *Autentisering av datakjelda* som vil seia at ein kan prova at kjelda er den ho hevdar å vera.
- *Datakonfidensialitet* som inneber at informasjonen ikkje er tilgjengeleg eller avdekka for uautoriserte.
- *Tidsgyldige meldingar* (*timeliness*) som mellom anna inneber at meldingar generert utanfor eit spesifisert tidsvindauge ikkje vert akseptert.

For å ivareta desse tenestene, vert det spesifisert ein del mekanismar, først og fremst i form av protokollar. Dei viktigaste mekanismane vert kort gjennomgått i dei følgjande avsnitta.

4.8.2.1 USM-tenester

Som andre modular har òg USM eit sett primitiv med tilhøyrande parametrar for kommunikasjon med andre modular, i første rekkje med meldingsprosesseringsmodulen. Primitiva er relatert til generering av utgåande og prosessering av innkomande meldingar. USM prosesserer verdiane for datasikringsparametrane i den endelege meldinga.

4.8.2.2 Autoritative og ikkje-autoritative SNMP Engines

For å kunna verna mot forseinking og repetisjon av meldingar, skal ein av entitetane som er involvert i kommunikasjonen, vera autoritativ. Dersom meldinga krev respons, er det mottakaren av meldinga som er autoritativ. Dersom meldinga ikkje krev respons, er det sendaren. Den autoritative entiteten kontrollerer tidsindikatorane i meldinga og kan setja i verk ulike funksjonar i tråd med desse.

4.8.2.3 Tidssynkronisering

Ein synkroniserer ved at den ikkje-autoritative entiteten lagrar ein kopi av tidsindikatorane som er mottekne frå autoritativ entitet og held dette opp mot lokal tid.

4.8.2.4 Autentisering

Autentiseringsprotokollane har to funksjonar. Dei skal sikra dataintegritet ved å kalkulera ei forkorting (*digest*) av meldinga og inkludera denne som del av meldinga. Mottakar vil då kunna avsløra om meldinga har vorte endra. Dei skal vidare sikra autentisering av kjelda. Kjelda nyttar difor ein hemmeleg nøkkel som vert lagt til meldinga før meldingsforkortinga vert utrekna. Hash-funksjonar vert nytta for å generera meldingsforkortinga. USM opnar for bruk av to alternative autentiseringsprotokollar:
- *Hash Message Authentication Code* (HMAC)-MD5-96. Protokollen er basert på ein 128 bitar lang autentiseringsnøkkel, og nyttar MD5 som underliggjande hash-funksjon. Ei 128 bitar lang meldingsforkorting vert produsert. Denne vert trunkert til 96 bit.
- *HMAC-Secure Hash Algorithm* (SHA)-*96*. Protokollen er basert på ein 160 bitar lang autentiseringsnøkkel, og nyttar SHA som underliggjande hash-funksjon. Ei 160 bitar lang

meldingsforkorting vert produsert. Denne vert òg trunkert til 96 bit.

4.8.2.5 Kryptering

USM nyttar *Cipher Block Chaining* (CBC)-modus av *Data Encryption Standard* (DES). I dette moduset er inngangsblokken ein klartekstblokk som vert kombinert med *foregåande krypterte blokk* ved hjelp av ein *XOR*-funksjon. For å kryptera den første klartekstblokken vert ein initialiseringsvektor nytta. Den hemmelege nøkkelen er 128 bitar lang. Av dei første 64 er 56 bitar nytta som ordinær DES-nøkkelen, medan dei siste 64 er ein *pre*initialiseringsvektor. Preinitialiseringsvektoren er òg kjent for mottakar og vert nytta til å rekna ut verdien for initialiseringsvektoren som skal vera unik for kvar melding. I meldinga vert det difor overført ein verdi som skal gjera det mogleg for mottakar å rekna ut initialiseringsvektoren som er nytta for denne meldinga.

4.8.2.6 Avdekkingsprosessen

USM krev at ein brukar ein spesifisert avdekkingsprosess for å oppnå tilstrekkeleg informasjon om SNMP-entitetar ein skal kommunisera med. Særleg må ikkje-autoritative entitetar vita identiteten til ein autoritativ entitet før kommunikasjonen kan halda fram. Dersom kommunikasjonen krev autentisering, må òg tidssynkronisering etablerast mellom entitetane.

4.8.2.7 Algoritme for nøkkellokalisering

Brukaren har hemmelege separate nøklar for kryptering og autentisering. Ein lokalisert nøkkel (*lokalized key*) er ein nøkkel som brukar deler med ein autoritativ entitet. Algoritmen genererer først ein nøkkel på grunnlag av passordet til brukaren. Nøkkelen går deretter saman med ID for aktuell SNMP-entitet gjennom ein hashfunksjon. På denne måten vil ein på grunnlag av eitt og same passord, kunna generera separate nøklar for kvar SNMPentitet brukaren skal kommunisera med.

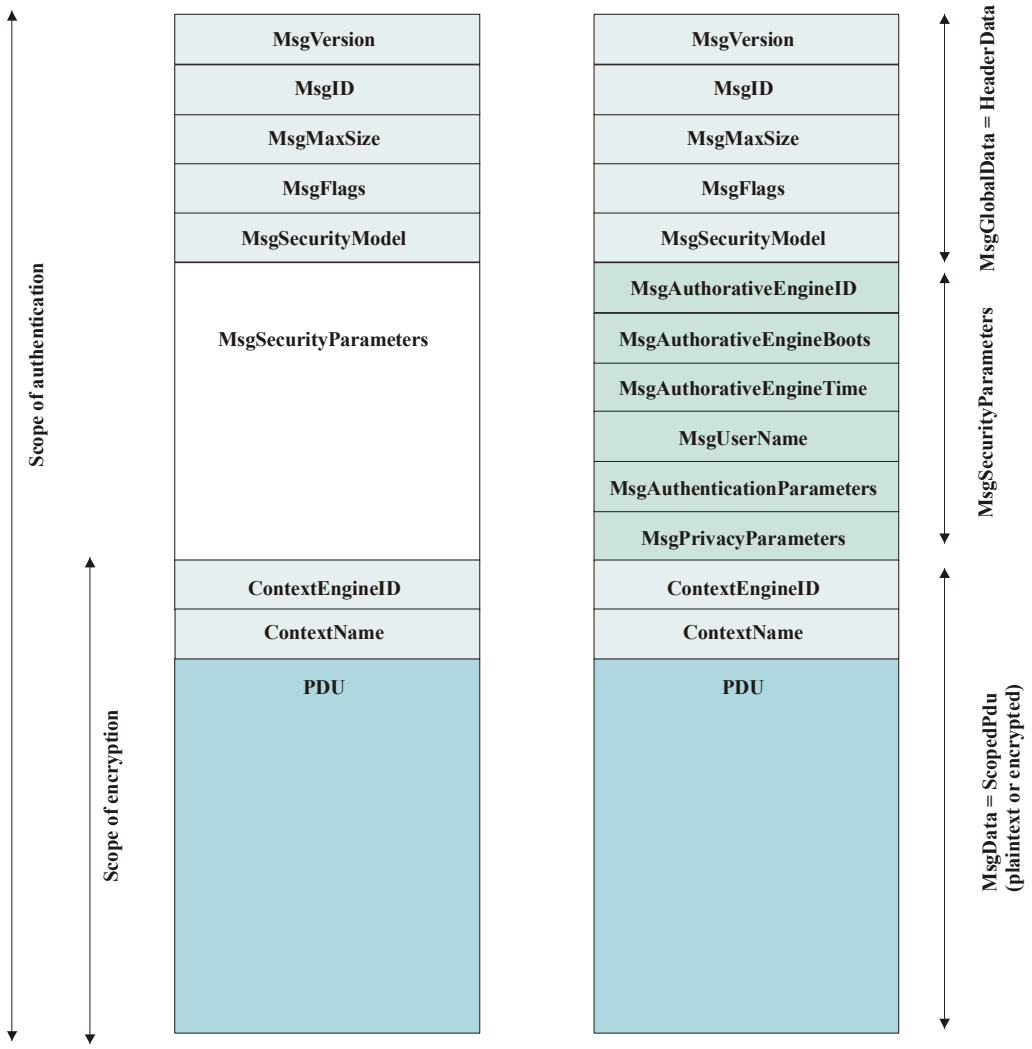
Dei hemmelege nøklane er ikkje lagra i MIBs og er difor ikkje tilgjengelege for SNMP. SNMP handterer ikkje overføring av nøklar. Oppdatering av nøklar vert derimot handtert.

4.8.2.8 USM MIB

Det er spesifisert ein MIB som inneheld ei objektgruppe, *usmUser*, med informasjon om lokale brukarar så vel som fjernbrukarar. Objekta innheld mellom anna informasjon om autentiserings- og krypteringsprototokollar for kvar brukar. Det finns òg objekt som vert nytta når brukaren oppdaterer nøklane. Sjølve nøklane er ikkje lagra i MIB

4.8.3 Meldingsformat

Figur 4.10 syner det nye meldingsformatet i SNMPv3.



Figur 4.10 Meldingsformat i SNMP v3. USM prosesserer verdiane MsgSecurityParameters

Ei melding er sett saman av:

- *Msg Version* som syner SNMP-versjonen.
- *MsgID* som er ein unik identifikator som vert nytta for å koordinera meldingane mellom to SNMP-entitetar.
- *MsgMaxSize*s om sendar nyttar til å informera mottakar om maksimums storleik for meldingar sendar kan handtera.
- *MsgFlags* som inneheld tre flagg. Det første syner om sendaren skal ha svar dersom spesifiserte vilkår tilseier det. Dei to siste syner kva sikringsnivå som er nytta for meldinga (autentisering og/eller kryptering).
- *MsgSecurityModel* som indikerer kva modell (i praksis SNMP-versjon) sendaren har nytta for å prosessera meldinga.

Dei ovanfornemnte felta utgjer meldingshovudet. Følgjande felt vert prosessert av USM dersom flagg for sikringsnivå tilseier det:

- *MsgAuthoritativeEngineID* som spesifiserer ID for den autoritative SNMP-entiteten.
- *MsgAuthoritativeEngineBoot* som spesifiserer kor mange gonger den autoritative SNMP-

entiteten har vorte initialisert etter første konfigurasjon.
- *MsgAuthoritativeEngineTime* som spesifiserer tida som har gått sidan *MsgAuthoritativeEngineBoot* vart inkrementert.
- *MsgUserName* som innheld namnet på brukaren som har sendt meldinga. Internt heiter parameteren *securityName*.
- *MsgAuthenticationParameters* som (dersom parameteren er sett) inneheld ein HMAC autentiseringskode for meldinga.
- *MsgPrivacyParameters* som (dersom parameteren er sett) inneheld ein verdi som skal gjera det mogleg for mottakar å kalkulera initialiseringsvektoren som er nytta i krypteringa av denne meldinga.

Dernest følgjer to felt som er relevante for tilgangskontrollen. (Omgrepet *kontekst* vert forklart i avsnitt 4.8.4.1.):
- *ContextEngineID* som for innkomande meldingar avgjer kva applikasjon som skal prosessera PDU. For utgåande meldingar vil verdien verta sett av applikasjonen som initierer meldinga.
- *ContextName* som unikt identifiserer ein kontekst.

Til slutt følgjer PDU som er spesifisert til å vera ein SNMPv2-PDU.

4.8.4 View-Based Access Control Model (VACM)

4.8.4.1 Element i modellen

Modellen er bygt opp rundt følgjande element:
- *Grupper*. Dette er ei samling av < *securityName*, *securityModel* > - tuplar, sjå avsnitt 4.8.3. Formålet er at brukarane på denne måten får tilgang til informasjonen som medlemar av ei gruppe.
- *Sikringsnivå*. Tilgangsrettane for ei gruppe kan definerast slik at tilgangen er avhengig av sikringsnivået til meldinga. Desse nivåa er: ingen autentisering ingen kryptering, autentisering ingen kryptering samt autentisering og kryptering.
- *Kontekst*. Ein kontekst er eit spesifisert subsett av informasjonen ein SNMP Entitet rår over.
- *MIB-utsnitt* (*views*) og *utsnittsfamiliar*. Ei gruppe har tilgang til ein kontekst via *utsnitt* av MIB. Eit utsnitt er eit detaljert subsett av informasjonen innan ein kontekst. Enkle utsnitt vil typisk vera *eitt* subtre av registreringstreet, medan meir komplekse utsnitt kan vera unionen av fleire slike subtre. Ein kan aggregera fleire subtre inn i ein og same struktur. Ein slik struktur vert kalla ein *utsnittsfamilie*. (Sidan objekt i ulike kolonnar i ein tabell ofte tilhøyrer ulike subtre i MIB, vil det vera naudsynt å setja saman mange subtre for å få tilgang til ein heil tabell).
- *Tilgangsreglar*. MIB-utsnitta kan grupperast ut frå kva type tilgang dei kan gi ei gruppe. Det finns tre typar: *lesing*, *skriving* og *notifisering*. Les-utsnittet vil tildømes spesifisera samlinga av objekt som gruppa kan lesa. Tilgangsretten for ei gruppe vil dermed vera gitt av tre MIB-utsnitt av gitt kontekst.

4.8.4.2 VACM MIB

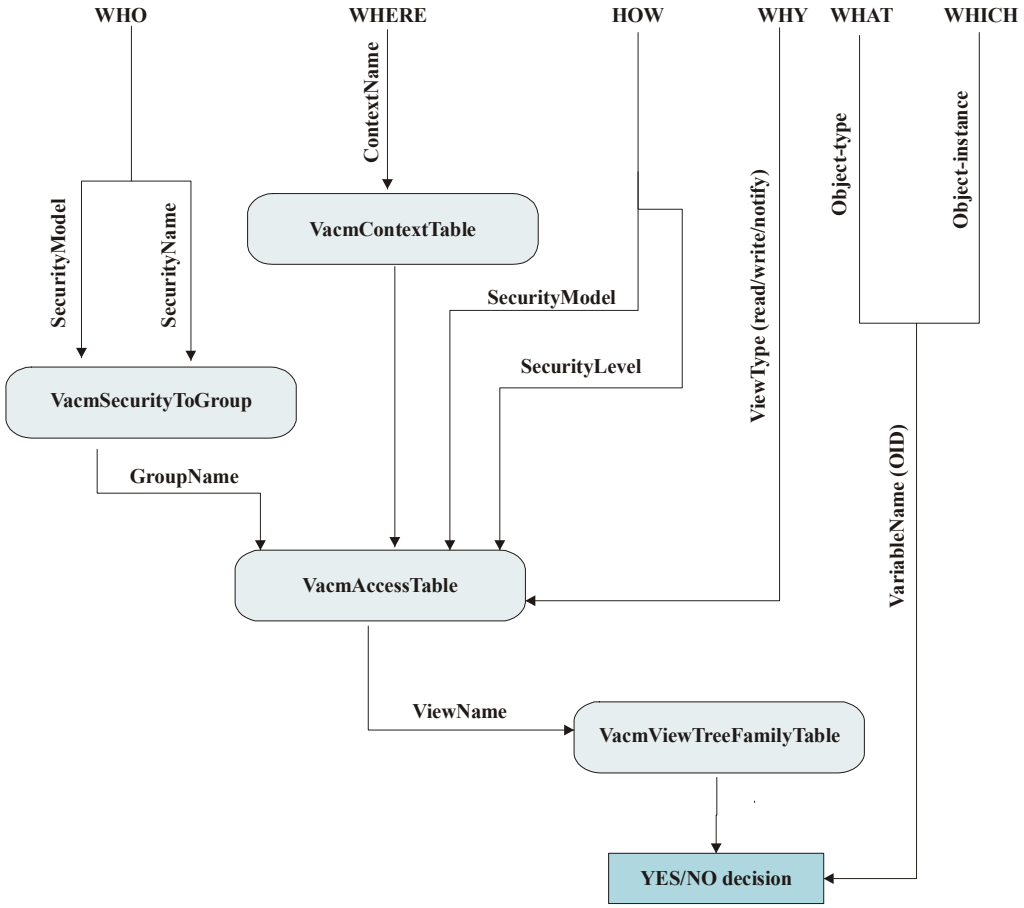
Det er definert ein MIB (i samsvar med SMIv2) med fire objektgrupper:
- *Local Contexts*. Objekta er ordna i ein tabell over alle kontekstar denne SNMP-entiteten har

definert.

- *Groups*. Objekta er ordna i ein tabell som tilordnar kombinasjonar av parametrane *securityName* og *securityModel* eit gruppenamn, sjå avsnitt 4.8.3.
- *Access Rights*. Objekta er ordna i ein tabell der tilgangsrettane (lesing, skriving, notifisering) til gruppene er definert for ein eller fleire kontekstar, eit eller fleire tilgangsnivå og for ein eller fleire sikringsmodellar. Tabellen spesifiserer relevante MIB-utsnitt.
- *MIB Views*. Objekta er ordna i ein tabell som spesifiserer dei ulike utsnittsfamiliane i SNMP-entiteten.

4.8.4.3 VACM-tenester

VACM-modulen har eitt primitiv med tilhøyrande parametrar. Tenesta som er bygd rundt dette primitivet, kontrollerer om brukaren er med i ei gruppe som har tilgang til å lesa eller modifisera ønska objektinstans. Det kan vera applikasjonane Command Responder eller Notification Originator som ber om tenesta. Figur 4.11 syner gangen.



Figur 4.11 Logikk i VACM (3)

Parametrane *SecurityModel* og *SecurityName* vert sendt til gruppe-tabellen i MIB for å finna namnet på gruppa brukaren er definert i. Gruppenamnet vert så sendt til tabellen over tilgangsrettar. Parameteren *ContextName* vert sendt til kontekst-tabellen for å stadfesta at konteksten finns. Det vert deretter sendt til tabellen over tilgangsrettar. Parametrane *SecurityModel* og *SecurityLevel* vert sendt til tabellen over tilgangsrettar

Ved hjelp av parametrane G*roupName*, C*ontextName*, *SecurityModel* og *SecurityLevel* vil ein i

tilgangstabellen finna tre aktuelle MIB-utsnitt. Parameteren *viewType* vert så nytta til å velja korrekt utsnittstype. Utsnittsnamnet vert så sendt til tabell over utsnittsfamiliar der ein finn aktuelt MIB-utsnitt. Utsnittet definerer variabelnamna som er med.

Parametrane *ObjectType* og ObjectInstance formar VaraiableName og dette vert kontrollert mot det utvalde MIB-utsnittet før tilgang vert gitt eller avslått.

4.9 Agent Extensibility (AgentX) Protocol

(47) foreslår ei utviding av den tradisjonelle agenten i SNMP-rammeverket. I AgentX-rammeverket er ein agent samansett av:

- Ein einskild *masteragent* som sender og mottek SNMP-meldingar i agentrolle slik denne er spesifisert i SNMP-rammeverket. Masteragenten har derimot lite eller ingen tilgang til informasjonen i MIBs.
- *Subagentar* som er ”verna” mot SNMP protokollmeldingar sidan desse vert prosessert av *masteragent*. Subagentane har tilgang til informasjonen. Dei interne operasjonane i ein subagent er usynlege for ein SNMP-entitet i manger-rolle.

Masteragent og subagentar kommuniserer via AgentX-protokollen.

Formålet med ”*extensible agents*” er å gjera det enklare å utvida ein node med nye objekt. Oppsettet er berre tilrådd når masteragent og subagentar er implementert på same vertsmaskin. Dette skuldast mellom anna at protokollen ikkje har sikringsmekanismar.

4.10 Oppsummering

Arkitekturen for SNMP er gjennomgått. Informasjonsmodellen skil seg sterkt frå OSI sin ved at han ikkje er objektorientert, og ved at objekta er skalare verdiar. Objekta er i stor grad ordna i todimensjonale konseptuelle tabellar. Objekttypar vert som i OSI, spesifisert ved hjelp av ASN.1. Det finns ingen funksjonsmodell. Kommunikasjonsmodellen er i versjon 1 og 2, berre sjølve SNMP-protokollen, medan versjon 3 introduserer ein meir heilskapleg og modulær arkitektur med til dømes definerte informasjonstenester mellom modulane.

SNMP v1 og v2 har svært svake sikringsmekanismar. Først i versjon 3 vert dette retta opp, først og fremst gjennom mekanismar for autentisering, kryptering og tilgangskontroll.

5 ANDRE ARKITEKTURAR

Datasystem har til nyleg i stor grad vore utvikla utan at ein har lagt særleg vekt på distribusjonsapektet. Store kommunikasjonsnett er ”av natur” *distribuerte system*. I tråd med dette synet er det utvikla arkitekturar for distribuert prosessering og for drift og styring av slike distribuerte system. Sidan dei underliggjande kommunikasjonsnetta i seg sjølv er distribuerte system, vil ein òg måtta utvikla drift- og styringssystema som distribuerte system.
Arkitekturane for distribuerte system er difor viktige ut frå to aspekt:

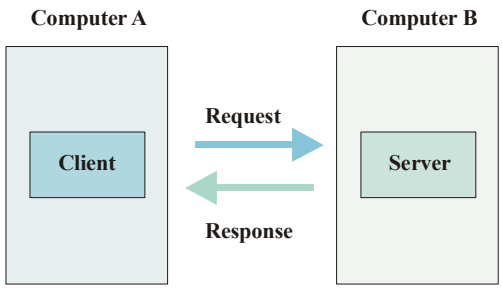
- Som arkitektur for det underliggjande kommunikasjonsnettet/systemet
- Som arkitektur for sjølve drift- og styringsnett/systemet

I dette avsnittet vil ein kort gå gjennom nokre av dei sentrale arkitekturane som har vorte utvikla i løpet av siste tiåret.

5.1 Distributed Computing Environment (DCE)

DCE var eit initiativ frå *Open Software Foundation* (OSF) og er eit rammeverk for samarbeid mellom opne system. DCE er ein definisjon av tenester og verktøy som skal støtta utvikling, bruk og administrasjon av distribuerte applikasjonar i heterogene datamaskinmiljø. Arbeidet med DCE spelte ein viktig rolle i utviklinga av klient-tenar-konseptet. DCE byggjer på tre viktige element:

- *Klient-tenar-modellen* vert nytta til å definera ein metode for å strukturera distribuerte applikasjonar.
- *Remote Procedure Call* (RPC) tilbyr ein mekanisme for direkte kommunikasjon mellom individuelle komponentar (prosessar) i ein distribuert applikasjon som køyrer på ulike system.
- *Shared Data Storage* gjer distribuert prosessering mogleg. Dei individuelle komponentane i ein distribuert applikasjon kommuniserer indirekte ved å bruka felles globalt tilgjengelege data.



Figur 5.1 Enkel klient – tenar modell (1)

Katalog (basert på X.500), synkronisering, datasikring og distribuert filsystem er sentrale DCE-tenester.

OSF har konstruert eit eige objektorientert drift- og styrings-system for DCE: *Distributed Management Environment* (DME). DME kan ved hjelp av “tilpassings-objekt” nytta både OSI og Internet-standardar. DME har ikkje lukkast i marknaden, og OSF har i staden gått inn for CORBA og konsepta frå *Object Management Group* (OMG) som støtte for DCE.

5.2 Open Distributed Processing (ODP)

ODP er utvikla i samarbeid mellom ISO og ITU-T, og er spesifisert i mellom anna ITU-T-rekommendasjonar, serien X.900- X.960. Det er utarbeidd ein referansemodell, RM-ODP.

Formålet med modellen er å støtta utviklinga av standardar for distribuerte informasjonsprosesseringstenester i store miljø med heterogene IT-ressursar og med mange organisatoriske domene. Modellen spesifiserer i seg sjølv ikkje nokon standard, men skaffar til vege eit konsept og ein terminologi for lettare å kunna utvikla standardar. Modellen kan difor sjåast som ein *meta-standard*. Sidan modellen er generisk, kan han nyttast i ulike domene. OSF, OMG og *Telecommunication Information Networking Architecture* (TINA)-konsortiet har nytta modellen.

Referansemodellen nyttar eit objektorientert konsept i spesifikasjonen av distribuerte system. Objektmodellen er allmenn. Alle komponentar i arkitekturen, til dømes prosessar og ressursar, kan sjåast som objekt. Objekta kan spesifiserast med fleire ulike grensesnitt.

For å handtera kompleksiteten i store distribuerte system, introduserer ein eit konsept der ein nyttar ulike synsvinklar (*viewpoints*) for å skildra systemet. Det vert definert fem slike synsvinklar:
- *Enterprise viewpoint* der formålet, omfanget og praksisen til systemet er fokusert.
- *Information viewpoint* der semantikken til informasjonen og informasjonsprosesseringa er fokusert.
- *Computational viewpoint* der ein vil mogleggjera distribusjon gjennom å funksjonelt dekomponera systemet til objekt med grensesnitt der interaksjonen mot andre objekt er spesifisert i forma av teneste-attributtar.
- *Engineering viewpoint* der mekanismar og funksjonar som støttar den distribuerte interaksjonen mellom objekta i systemet, er fokusert.
- *Technology viewpoint* som omhandlar val av teknologi i systemet.

Det er eit viktig prinsipp at ODP skal vera transparent med omsyn til:
- *Tilgang*. Dette vil seia at eit objekt ikkje ”ser” ulike meldingsformat og protokollar men eit standardisert grensesnitt.
- *Lokasjon*. Dette vil seia at eit objekt ikkje ”ser” lokasjonen til eit samarbeidande objekt.
- *Migrasjon*. Dette vil seia at eit objekt kan relokaliserast utan at interaksjonen med andre objekt vert avbrote.
- *Samanfall i tid*. Dette vil seia at fleire objekt *samstundes* kan arbeida mot *eitt* objekt, men slik at konsistensen til objekta vert oppretthalden.
- *Feil*. Dette vil seia at eit objekt ikkje ”ser” feil og evt gjenvinningsaktivitetar i eit anna objekt.
- *Replikasjon*. Dette vil seia at eit objekt ikkje ”ser” at eit anna vert replikert.

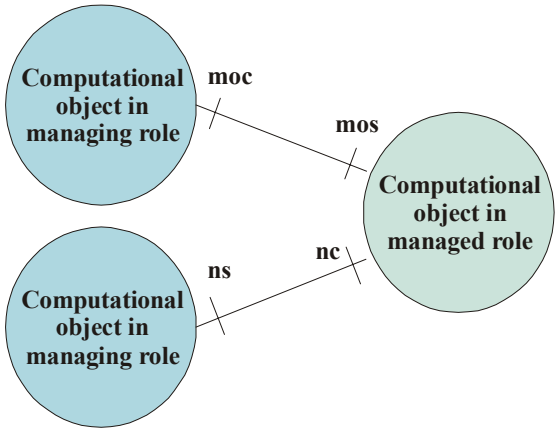
(50) gir ei rekkje døme på korleis ulike deler av eit telekommunikasjonsnett kan modellerast ved hjelp av RM-ODP.

5.3 Open Distributed Management Architecture (ODMA)

På bakgrunn av RM-ODP er det utarbeidd fleire standardar som omhandlar ulike aspekt ved konstruksjon av, og operasjonar i, distribuerte system. ODMA (11) skaffar til vege ein slik spesialisert arkitektur. Arkitekturen skal gjera det mogleg å spesifisera og utvikla *både* drift- og styringsfunksjonen som distribuert applikasjon *og* drift og styring av allmene distribuerte applikasjonar. At ODMA er ein referansemodell for *distribuert* drift og styring av distribuerte

ressursar, system og applikasjonar inneber:
- Sjølve drift- og styringsaktiviteten er distribuert.
- Applikasjonane som skal styrast er distribuerte.
- Ressursane som skal styrast kan vera distribuerte.

(11) skildrar korleis objekt relatert til drift og styring skal spesifiserast. Desse objekta skal ha eit eller fleire *drift- og styringsgrensesnitt*, sjå omtalen av *Computational viewpoint* i forrige avsnitt. Det er tre ulike typar grensesnitt. Desse kan ha ulike rollar; klient eller tenar. Figur 5.2 illustrerer grensesnitta for operasjonar og notifikasjonar



Interface role	Type of ODMA computational operation interface
managing client role	management-operation client (moc) interface
managing server role	notification server (ns) interface
managed client role	notification client (nc) interface
managed server role	management-operation server (mos) interface

Figur 5.2 Samanhengen mellom rollar og typar for operasjonar og notifikasjonar (11)

OSI-konseptet for drift og styring, som vart gjennomgått i kapittel 2, kan i stor grad innkorporerast og gjenbrukast i ODMA. Ein vil då kunna bruka eksisterande standarder saman med teknikkar og mekanismar som er implementert etter ODP-prinsipp. (11) skildrar korleis dette kan gjerast. Støtte for ODMA vert slik eit supplement til OSI-arkitekturen.

Å bruka dei eksisterande drift- og styringsstandardane vert dermed *ein* mogleg måte å realisera ODMA. Ei anna tilnærming, som nødvendigvis ikkje står i motsetning til den første, er å bruka CORBA. (11) skildrar òg korleis dette kan gjerast.

ODMA er tilrådd arkitektur når sjølve drift- og styringsaktiviteten er distribuert og når applikasjonane eller ressursane som skal styrast, er distribuerte.

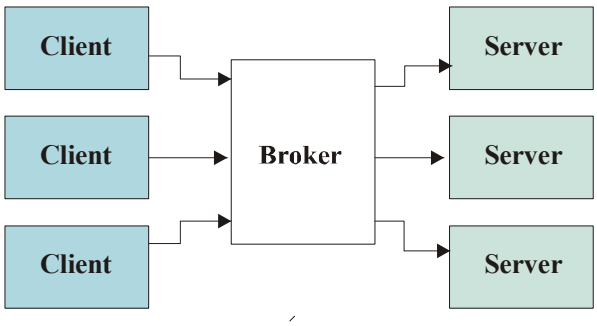
5.4 Common Object Request Broker Architecture (CORBA)

5.4.1 Generelt om CORBA

CORBA er utvikla av OMG, og kan med nokre avgrensingar sjåast som ei instansiering av RM-ODP sin objektmodell og er det første kommersielle produktet basert på ODP-spesifikasjonane.

CORBA er ein infrastruktur for distribuert prosessering og representerer ei utviding av klient-tenar-prinsippet ein kjenner frå DCE, sjå avsnitt 5.1:

- Større fleksibilitet i tråd med at ein introduserer ein formidlarfunksjon mellom klient og tenar. I dei tradisjonelle klient-tenar-systema har klient og tenar hatt definerte 1:m-relasjonar. I RPC må klienten adressera tenaren, sjå asvnitt 5.1. *Object Request Broker* (ORB) er ein infrastrukturkomponent som formidlar interaksjon mellom klientapplikasjonar som krev tenester og tenarapplikasjonar som skaffar til vege desse tenestene. ORB etablerer klient-tenar- relasjonar mellom objekt. Når ORB vert nytta kan eit klientobjekt setja i gang ein prosess på eit tenarobjekt, uavhengig av kvar objekta er lokalisert i nettet. ORB mottek ein forespørsel og er ansvarleg for å finna rette tenarobjektet, overføra parametrar, setja i gang prosessen oh levera resultata.



Figur 5.3 Kommunikasjon med broker (1)

- Dei kommuniserande partane er likeverdige objekt.
- Spesifikasjonen er objekt-orientert, noko som inneber data-abstraksjon, innkapsling, arv og polymorfi. Dette forenklar modelleringsprosessen og tillet gjenbruk av objektspesifikasjonane.

CORBA skal i prinsippet kunna støtta alle typar distribuerte sysem og er difor ikkje spesielt innretta på kommunikasjonsnett og styring av slike. Informasjonsmodellen er følgjeleg allmenn. Førebels er modellen lite spesialisert for drift og styring av kommunikasjonsnett samanlikna med modellane i arkitekturane som vart gjennomgått i kapitla 2, 3 og 4.

Det er definert *tenester* som sikrar den grunnleggjande funksjonaliteten som trengs for å kunna nytta eit CORBA-basert system. Døme på ei slik teneste er *event service* som støttar kommunikasjon mellom objekt slik at dei kan senda asynkrone meldingar (utan forespørsel). Det er vidare definert *fasilitetar*. Dette er tenester som skal vera tilgjengeleg for alle applikasjonar, til dømes støtte for informasjonsmodellering.

I tillegg til å støtta kommunikasjon mellom objekt via ORBs, definerer CORBA òg protokollar for kommunikasjon mellom ulike ORBs i eit heterogent miljø:

- *General Inter-ORB protocol* (GIOP) spesifiserer syntaks og semantikk for meldingsutvekslinga mellom ORBs. Fleire PDU-typar er definert. Protokollen er laga slik at han kan nyttast over ein kvar tilknytingsorientert transportprotokoll.
- *Internet Inter-ORB Protocol* (IIOP) er eigentleg ”GIOP-over-TCP” og spesifiserer ei avbildning mellom GIOP og TCP. GIOP-PDUs vert lagt inn i TCP-PDUs.

- *Environment-Specific Inter-ORB Protocols* (ESIOPs) er laga for kommunikasjon over andre protokollar/protokollstakkar.

5.4.2 Object Management Architecture (OMA)

CORBA er kjernen i OMA, som òg vert utvikla av OMG. Dei ulike submodellane i OMA er synt i tabell 5.1.

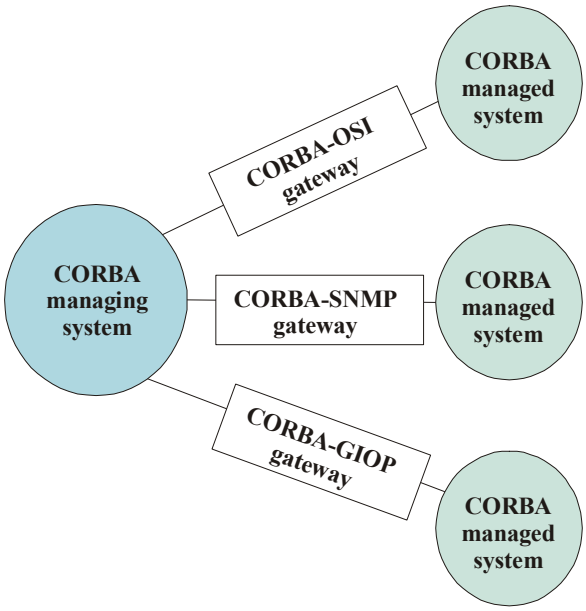
Object Management Architecture			
Functional Model	**Information Model**	**Communication Model**	**Organizational Model**
Domain Interfaces	CORBA object model	CORBA ORBs	Distributed objects
CORBA Facilities Common Mgmt. Facilities	Interface Definition Language (IDL)	Inter-ORB protocols (GIOP, IIOP, ESIOPs)	CORBA Interoperability Architecture
CORBA Services			

Tabell 5.1 OMA submodellar (1)

I denne arkitekturen er det førebels ikkje definert nokon drift-og styringsinformasjon som kan konkurrera med OSI og SNMP-MIBs. Derimot er det som ei mellombels løysing utvikla standardiserte translasjonsprosedyrar slik at dei ovanfornemnte informasjonsbasane kan nyttast i CORBA-miljø.

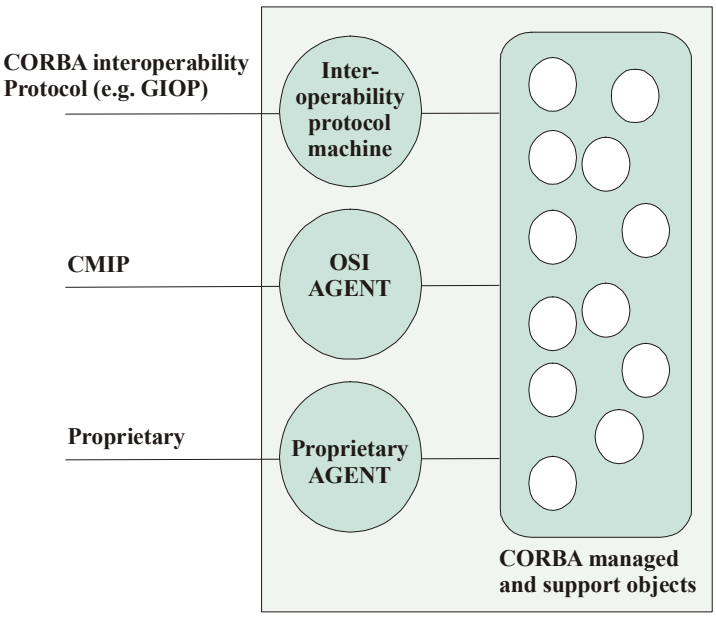
Særleg på teneste- og forretningslaget i TMN ser det ut til at CORBA-baserte applikasjonar vil verta nytta.

Som tidlegare nemnt kan CORBA nyttast i realiseringa av ODMA. Figurane 5.4 og 5.5 syner korleis ein tenkjer seg dette.



Figur 5.4 CORBA-basert styrande system (11)

Figuren over syner korleis eit CORBA-basert *styrande* system kan kommunisera med ulike typar agentar med ulike protokollar via *gateways*.



Figur 5.5 CORBA-basert styrt system (11)

Figuren over syner korleis eit CORBA-basert *styrt* system kan kommunisera med ulike styrande system som støtter ulike protokoller. Dei styrte objekta kan til dømes vera definert i samsvar med ISO-standarden, sjå avsnitt 2.4, men kan vera implementert som CORBA-objekt. Ein vil då kunna få tilgang til objekta både via CMIP/OSI-agent og via GIOP.

5.5 Telecommunication Information Networking Architecture (TINA)

TINA er utvikla av *TINA Consortium* (TINA-C), ei samanslutning av nettoperatørar og leverandørar av telekommunikasjon og IT.

Formålet med TINA var å skaffa til vege eit rammeverk for utvikling av framtidige telekommunikasjonsnett. Arkitekturen er basert på RM-ODP og er følgjeleg modellert ut frå dei fem synsvinklane som var omtalt i avsnitt 5.2. TINA har vorte spesialisert for å møta dei særeigne krava innan telekommunikasjonsdomenet. Vidare er TINA laga slik at eksisterande arkitekturar for til dømes Intelligente Nett (IN) og TMN skal kunna innkorporerast.

TINA er sett saman av fire blokkar (1):

- *Arkitektur for distribuert prosesseringsmiljø*. Denne blokken skal leggja grunnlaget for dei tre andre, og formålet er å gi ei homogen plattform for ulike distribuerte applikasjonar. *Distributed Processing Environment* (DPE) er på mange vis kjernen i TINA. DPE skal implementerast med CORBA som utgangspunkt. Objekt og grensesnitt er spesifisert. Ein tenkjer seg eit eige logisk nett, ein mellomvareinfrastruktur, for informasjonstransport mellom DPE-nodane.
- *Arkitektur for nettressursar*. Denne blokken definerer konsept for styring av transportnett. I modelleringa av nettressursar skil ein ikkje mellom *control* og *management* slik det ofte er gjort i eksisterande nett (i form av *control functions* og *management functions*.) Det er laga ein informasjonsmodell for nettressursar, *Network Resource Information Model* (NRIM).

Modellen definerer notasjonar for nettelement og koplingar. Notasjonane er uavhengige av transmisjons- og svitsjeteknologi. Basert på desse notasjonane er det laga ein generisk nettarkitektur som skildrar korleis dei individuelle elementa i eit nett er relatert, den innbyrdes konnektiviteten og korleis dei kan konfigurerast for å utgjera ein ende-til-ende kommunikasjonssti. Det vert òg definert ein generisk komponent som styrer/kontrollerer denne stien. Ein baserer seg på at eksisterande informasjonsmodellar frå TMN framleis skal brukast. Dette gjeld både den generiske og dei teknologispesifikke som er utvikla på grunnlag av denne. NRIM er eit viktig tillegg til desse modellane:

- Det vert definert ei rekkje nye fragment, sjå avsnitt 3.3.1.
- Det er særleg viktig at modellen skal danna eit grunnlag for drift og styring av *tenester*. I tråd med dette kategoriserer ein objekta med omsyn til abstraksjonsnivå:
  - Objekt som inneheld detaljert informasjon om nettstrukturen medan den underliggjande teknologien vert skjult
  - Objekt som inneheld eit samandrag av eigenskapane til eit (sub)nett. Informasjonen er til ein viss grad aggregert frå den første kategorien.
  - Objekt som er dedikert for ulike telekommunikasjonstenester.

- *Arkitektur for tenester*. Denne blokken definerer konsept for tenesteutvikling i eit TINA-miljø. Tenesteapplikasjonane ligg oppå det CORBA-baserte prosesseringsmiljøet. Dette vil seia at ein på tenestelaget ikkje ”ser” kompleksiteten og heterogeniteten i sjølve distribusjonen, til dømes dei ulike datarepresentasjonane, kvar objekta er lokalisert osv.
- *Arkitektur for drift og styring*. Denne blokken skaffar eit sett av generiske drift-og styringsprinsipp for dei tre andre blokkane. Arkitekturen dekkar to felt: endesystem og telekommunikasjon. Sistnemnte er primært basert på eksisterande prinsipp og standardar for OSI og TMN. Dei ”logiske laga” frå den funksjonelle arkitekturen for TMN, sjå avsnitt 3.2.5, vert i særleg grad trekt inn. TINA omhandlar element-, nett- og tenestelaget. Planen er å gradvis utvikla eksisterande TMN-system til å kunna operera i distribuerte TINA-miljø.

5.6 Web-basert drift og styring

Særleg i samband med intranett har det vore eit sterkt ynskje om å kunna nytta web-lesarar som grafisk brukargrensesnitt for drift og styring. Fordelen med dette er mellom anna at grensesnittet i praksis vert plattformuavhengig. Det er to ulike måtar å nytta web-lesarar som konsoll for drift og styring (1):

- Lesar kommuniserer direkte med dei styrte ressursane ved hjelp av *HyperText Transfer Protocol* (HTTP).
- Lesar kommuniserer ved hjelp av HTTP med ei drift og styringsplattform som igjen kommuniserer med dei styrte ressursane over ein ordinær drift- og styringsprotokoll, til dømes SNMP.

Mange leverandørar av drift- og styringssystem arbeider med web-baserte løysingar. I det følgjande vert to av arkitekturane kort presentert.

5.6.1 Java Management API (JMAPI)

Konseptet deler drift-og styringssystema inn i to komponentar: konsoll og plattform. Desse komponentane kan på mange vis samanliknast med funksjonsblokkane WSF og OSF i TMN. Arkitekturen er sett saman av fire element:

- *Styrt objekt-tenar*. Kjernen er ein komponent som tillet samarbeidande applikasjonar å oppretta, manipulera og sletta objekt. Komponenten har grensesnitt mot ein database for

styrte objekt, mot agentar som er implementert i Java over *Remote Method Invocation* (RMI) og mot SNMP-agentar over SNMP. Komponenten lagrar kode for å kunna implementera drift-og styringsapplikasjonar på agentar. Denne koden er som regel applets som er tilgjengelege frå ein HTTP-tenar.
- *Drift- og styringskonsoll*. Dette er ein web-lesar med ei integrert Java-maskin og utgjer brukargrensesnsitt for manipulasjon av styrte objekt osv.
- *Drift- og styringsagentar*. Desse er implementert i Java og kallast *appliances*. Koden kan vera integrert i agenten, eller lastast frå styrt objekt-tenaren.
- *Grensesnitt mot SNMP-agentar*.

Funksjonsmodellen er førebels lite utvikla. Det finns ein modell for handtering av hendingar.

JMAPI informasjonsmodell er ei vidareutvikling av den objektorienterte *Java Object Model*. Modellen innkorporerer dei vanlege prinsippa for objekt-orientering.

JMAPI kommunikasjonsmodell er basert på tre ulike kommunikasjonsprotokollar:
- HTTP er nytta for å lasta *Java applets* frå ein styrt objekt-tenar til drift- og styringskonsollet.
- RMI vert nytta for kommunikasjon mellom drift- og styringsapplikasjonane og dei styrte objekta i objekt-tenaren. Desse objekta nyttar vidare RMI i kommunikasjonen med agentane.
- SNMP vert nytta mot SNMP-agentar.

Det finns ein modell for spesifikasjon av notifikasjonar.

5.6.2 Web-Based Enterprise Management (WBEM)

Etter å ha mislukkast med den nye drift- og styringsprotokollen *Hypermedia Management Protocol* (HMMP), har konsortiet bak WBEM overlate det vidare arbeidet til *Distributed Management Tast Force* (DMTF).

Medan Java-arkitekturen legg vekt på å utvikla agentar og på gjenbruk av desse, set WBEM-arkitekturen informasjonsaspektet i fokus. Arkitekturen har følgjande deler (1):
- *Common Information Model* (CIM) som er ein objektorientert modell for å skildra drift- og styringsinformasjon.
- *Managed Object Format* (MOF) som er ein syntaksspesifikasjon for styrte objekt som har vorte modellert ved bruk av CIM. MOF kan samanliknast med OSI GDMO og SNMP SMI.
- *Ein spesifikasjon for å representera CIM i Extensible Markup Language* (XML). Dette gjer at einkvar web-lesar som støttar XML, kan nyttast som drift- og styringskonsoll.

Dei viktigaste elementa i CIM er:
- *Objektklassar* som er eit sett med objekt med same eigenskapar.
- *Metodar* som definerer operasjonar på objekta.
- *Kvalifiserarar* som spesifiserer tilleggseigenskapar for objektklassane eller for eigenskapane og metodane for ein klasse.
- *Assosiasjonar* som representerer relasjonar mellom objekt.
- *Referansar* som er eigenskapar med referanse til andre objekt.
- *Hendingsklassar* som vert nytta for å definera notifikasjonar som skal genererast av ein triggarmekanisme.
- *Skjema* som vert nytta til å gruppera element for administrative formål.

Kommunikasjonsmodellen er basert på HTTP som drift- og styringsprotokoll. XML er nytta for å kunna representera strukturerte data (CIM-objekt) som tekst.

5.7 Oppsummering

Nokre arkitekturar for distribuerte system er kort presentert. Med unnatak av DCE, som er den eldste av desse, ser ein følgjande fellestrekk:

- Dei er basert på objektorienterte konsept.
- Dei er basert på generiske informasjonsmodellar.
- Dei har som viktig målsetting å skaffa til vege ei plattform for utvikling, produksjon og drift/styring av generelle tenester så vel som telekommunikasjonstenester. Denne plattforma skal mellom anna kunna skjula kompleksiteten i det underliggjande kommunikasjonsnettet.
- Sidan dei er utvikla parallellt i tid, har det vore ei gjensidig påvirkning på, og gjenbruk av, konsept og løysingar. Nokre av resultata er standardisert gjennom ISO og ITU-T.

Drift- og styringskonsepta fokuserer i tillegg:

- Integrasjon av eksisterande drift- og styringsstandardar; TMN og SNMP.
- Drift og styring av distribuerte tenester og applikasjonar.
- Handering av mange og ulike domene.

Arkitekturane syner korleis dei sentrale aktørane innan telekommunikasjon tenkjer seg framtidige kommunikasjonsnett. Desse arkitekturane vil sjølvsagt leggja premiss for dei systema som skal styra desse kommunikasjonsnetta. Arkitekturen for kommunikasjonsnettet vil leggja føringar for informasjonsmodell, funksjonalitet og kommunikasjonsløysingar i drift- og styringssystema. I tillegg vil prinsipp og konsept frå arkitekturane for allmenne kommunikasjonsnett kunna brukast i utviklinga av nye drift-og styringssystem, sidan desse òg er distribuerte system.

Desse arkitekturane er utvikla i løpet av 90-talet. Det er i samband med denne rapporten ikkje undersøkt i kor stor grad konsept og prinsipp er implementert. Dette gjeld til dømes den planlagte migrasjonen frå eksisterande TMN-løysingar til TINA, bruk av CORBA i store kommunikasjonsnett osv.

6 UTVIKLINGSTRENDAR FOR DRIFT- OG STYRINGSKONSEPT

I dette kapittelet ser ein på ulike trendar for drift- og styringssystem. (49) gjennomgår generelle trendar for kommunikasjonssystem, og desse trendane vil sjølvsagt vera ei viktig råme for drift- og styringssystema. Fiberteknologi vil stå i sentrum i kommunikasjonsinfrastrukturen. Transportnettet vil i løpet av relativt kort tid gradvis verta heil-optisk. I første omgang vil nettet vera linjesvitsja, i neste truleg pakkesvitsja. Dette synspunktet er lite kontroversiellt.

Ein har likevel siste tiåret sett at IKT-marknaden er vanskeleg å spå. Dette gjeld særleg for tenesteproduksjon, som truleg òg i framtida vil ha ein mykje kortare tidshorisont enn til dømes teknologiutvikling for transportinfrastruktur. At transportnettet vert heil-optisk, vil seia at ein skiftar ut elektroniske komponentar med optiske. Det ligg *ikkje* i korta at dette i seg sjølv vil medføra at teknologien i transportnettet vert meir homogen. Gammal og ny teknologi vil eksistera side om side, og dette vil måtta medføra hybride løysingar og ad hoc-løysingar både i trafikknetta og i drift- og styringssystema.

I dag dreier drift og styring av kommunikasjonssystem seg i første rekkje om det fysiske/logiske nettet. Dei styrte entitetane er først og fremst fysisk *utstyr* og fysiske/logiske *linjer og koplingar*. I nyare arkitekturar ser ein kommunikasjonssystema som distribuerte dataprosesseringssystem, og ein ynskjer å mogleggjera drift og styring av distribuerte *applikasjonar* og *tenester* i slike kommunikasjonssystem. I tillegg finns det ein klar trend i retning av at kunden skal kunna styra gitte parametrar vedrørande eigen kommunikasjon og eigne tenester. Dette gjer at ein i utviklinga av framtidige drift- og styringsystem vil måtta leggja stor vekt på

- *Multidomene*. Drift- og styringsystema må kunna handtera mange og ulike domene med ulike krav til tenestekvalitet, datasikring osv. Grensesnitta mellom ulike domene, til dømes mellom operatørar og ulike tenestetilbydarar, vil få særleg fokus. Drift- og styringssystema har hittil handtert vertikal kommunikasjon i hierarkiske manager-agentstrukturar. Systema må i framtida òg kunna handtera horisontal trafikk mellom ulike sidestilte domene. Ulike kommunikasjonsteknologiar kan òg representera ulike domene. I eit framtidig "IP-over-WDM" vil ein til dømes kunna ha eit WDM-domene og eit IP-domene styrt av ulike operatørar.
- *Høgare endringsfrekvens*. Drift- og styringsystema må kunna handtera reelle, virtuelle og gjerne mobile organisasjonsstrukturar i rask endring samt samhandlinga mellom dei ulike organisasjonane. Dette gjeld organisasjonane til nettoperatørane og tenesteleverandørane så vel som kundeorganisasjonane. Konfigurasjonar og topologiar vil endrast langt oftare enn tilfelle har vore i dei store telekommunikasjonsnetta til no.
- *Datasikring*. At Internett i større og større grad vert nytta for informasjon som er kritisk for verksemdene, til dømes via virtuelle private nett, vil medføra at krava til datasikring aukar. Dette vil føra med seg auka krav til drift- og styringssytema. Når kundar, tenesteleverandørar og operatørar alle skal setja parametrar for eitt og same nett-element, eller ein og same applikasjon, vil dette vera ei stor utfordring for datasikringa i drift- og styringssystema.

Det er ikkje opplagt at eksisterande standardar og arkitekturar vil kunna møta desse krava. På kort sikt er det derimot lite som tilseier at dominansen som SNMP- og TMN-baserte system har på sine respektive felt vil svekkast.

Den framtidige utviklinga av SNMP verkar i dag nokså uklar. SNMPv2 har ikkje vore nokon suksess i marknaden. SNMPv3 har ein sterk konkurrent i IPsec (49). Det kan sjå ut til at SNMPv1 vil leva vidare enno ei stund, men vil møta store utfordringar med omsyn til:

- *Datasikring*. Kor vidt den mangelfulle datasikringa vert eit problem, er sjølvsagt avhengig av kva marknaden "tåler".
- *Informasjonsmodellen*. Den data-type-orienterte informasjonsmodellen framstår som ei hindring andsynes framtidige utfordringar for drift- og styringssystem. Det er til dømes svært vanskeleg å byggja distribuerte løysingar utan objekt-orienterte konsept og metodar. CIM, sjå avsnitt 5.6.2, kan verta ein utfordrar for SNMP SMI.

TMN vil truleg ta SNMP opp i seg, og den framtidige utviklinga av TMN vil etter alt å døma gå i retning TINA. Bruken av mellomvareteknologi vil kunna gjera det mogleg å byggja distribuerte drift- og styringssystem. Likeeins vil denne teknologien kunna skjula kompleksiteten i nettet slik at ein raskt kan utvikla applikasjonar på tenestelaget. Ei stor utfordring for TMN-systema er til dømes å kunna handtera raske endringar i nett-topologi og konfigurasjonar.

På kort og midlare sikt kan det sjå ut til at SNMP-baserte system framleis vil vera dominerande i lokalnett og for enkle driftsfunksjonar (overvåking av IP-trafikk), medan TMN-baserte system framleis vil dominera i komplekse nett og for komplekse drift- og styringsfunksjonar. På den andre sida vil ein i desse kommunikasjonsnetta ha eit stort og aukande innslag av utstyr som tradisjonelt har vore overvaka av SNMP-system, først og fremst IP-rutarar. Difor må TMN-systema på kort sikt vera i stand til å integrera nettelement som er basert på SNMP-drift. Integrasjonen mellom dei to konsepta vert omtalt i avsnitt 6.2.3

Ein ser ein klar trend i retning av at drift- og styringsfunksjonar vert flytta ned mot nettelementet for å gjera desse sjølvstyrte. Ein tenkjer seg at netta i framtida i stor grad vil vera "sjølvregulerande" med omsyn til trafikklast og overlevingsevne. I denne samanhengen er proaktiv feilhandtering viktig.

6.1 Distribuert drift og styring

Mellomvareteknologi hevar, som før nemnt, abstraksjonsnivået for utvikling av distribuerte applikasjonar. Produkta, som til dømes CORBA, *Component Object Model* (COM), *Distributed COM* (DCOM) og JAVA RMI er utvikla for distribuerte system allment, ikkje for kommunikasjonssystem spesielt. Denne teknologien har nok i stor grad vore sett på som eit universalmiddel for å få til interoperabilitet og konnektivitet i heterogene systemmiljø. Dette er sjølvsagt ikkje tilfelle. Det ligg store, og førebels uløyste, utfordringar i å byggja dei effektive storskala globale distribuerte applikasjonane ein i dag ser føre seg trengs for til dømes "e-business" osv. Nokre av desse utfordringane er (52):

- *Skalering*. Dette dreier seg i første rekkje om kor godt systemet evnar å ta opp i seg nye applikasjonar, brukarar og nodar utan at den spesifiserte tenestekvaliteten går tapt.
- *Mobilitet*. Dette gjeld evna til å handtera både mobile brukarar og mobilt terminalutstyr.
- *Drift og styring*. Dette gjeld evna til å styra og vedlikehalda objekta i det distribuerte systemet. Lokalisering av objekt i mobile miljø vert ei særleg utfordring.

I kommunikasjonssystem skal mellomvareteknologi sørgja for ein software infrastruktur som gjer at *tenesteelement*, gjerne frå ulike tenesteleverandørar, skal kunna kommunisera med *nettelement*. Samstundes skal teknologien sørgja for at kompleksiteten i nettlaget vert skjult for laga over. Det vil vera viktig at tenesteapplikasjonar kan utviklast og endrast raskt.

I kommunikasjonssystem med distribuert dataprosessering, vil drift- og styringssystema i seg sjølv vera distribuerte applikasjonar. Alle nye arkitekturar vektlegg dette. Det tradisjonelle statiske manager-agent-konseptet vil måtta vika for distribuerte og autonome managerprosessar. Nokre av desse problemstillingane vert synleggjort i arbeidet med drift- og styringskonsept for heil-optiske transportnett.

6.1.1 Drift og styring av heil-optiske transportnett

Heil-optiske svitsja transportnett er gode døme på distribuerte prosesseringsmiljø. Dei ulike nodane i nettet, særleg dei optiske krysskoplarane, må til dømes ha tilgang til mykje informasjon om lokal topologi og linkstatus for å kunna handtera dynamiske rutingsalgoritmar og automatisk krysskopling/svitsjing.

Optiske signal er karakterisert av bølgjelengda og kan prosesserast pr bølgjelengde eller som ei bølgjelengdedelt multipleksa gruppe av bølgjelengder. ITU-T, som har standardisert infrastrukturen til optiske transportnett (OTN), deler nettet inn i tre lag (26):

- *Optisk kanallag*. Dette laget sørgjer for å transportera digitale klientsignal gjennom ein optisk kanal (ende-til-ende) mellom aksesspunkt. Klientar kan vera SDH, STM, PDH, ATM osv. Karakteristisk informasjon på dette laget er klientdata saman med kanaladministrative data vedrørande terminering av kanalen. Transporteininga kallast *Optical Transport Unit* (OTU). Kvar kanal har ei definert bandbreidde. Kanalen vil kunna rutast over fleire subnett. På dette laget vil det såleis vera stor fleksibilitet med omsyn til ruting.
- *Optisk multiplekssekjonslag*. Dette laget sørgjer for å transportera dei optiske kanalane gjennom ein optisk multiplekssekjon mellom aksesspunkt. Karakteristisk informasjon på dette laget er ein straum som er sett saman av *n* optiske kanalar som til saman har ei definert aggregert bandbreidde. I tillegg kjem multipleksadministrative data vedrørande terminering av seksjonen. Transporteininga kallast *Optical Transport Unit Group of order n* (OTUG*n*). Det er ikkje definert subnett for dette laget, og det vil ikkje vera nokon fleksibilitet med omsyn til ruting.
- *Optisk transmisjonsseksjonslag*. Dette laget sørgjer for å transportera ein optisk multiplekssekjon gjennom ein optisk transmisjonsseksjon mellom aksesspunkt. Ein optisk transmisjonsseksjon transporterer ein einskild instans av mulitplekssekjon. Det er såleis eit ein-til-ein-forhold mellom dette laget og laget over. Karakteristisk informasjon på dette laget er informasjonen frå laget over og transmisjonssadministrative data vedrørande terminering av seksjonen. Transporteininga kallast *Optical Transport Module of order n* (OTM*n*). Seksjonen definerer eit fysisk grensesnitt med optiske parametrar.

Samband på alle lag er bi- eller unidireksjonale og har punkt-til-punkt eller punkt-til-multipunkt terminering i aksesspunkt. Under desse nettlaga kjem så lag for fysisk medium som definerer fibertype. Dette laget har ingen aktive komponentar.

Når det gjeld drift og styring av desse tre laga, gjenstår enno mykje spesifikasjonsarbeid. Det er

spesifisert nokre grunnleggjande krav innan dei tre funksjonsområda *feilhandtering*, *konfigurasjon* og *yting*. Funksjonalitet som skal handterast er til dømes:

- *Kontinuiteten* på einskildsamband på alle tenarlag skal overvakast. Tap av kontinuitet skal meddelast klienlaget. Tap av kontinuitet kan skuldast feil på fiber eller optiske komponentar.
- *Konnektiviteten* i rutinga skal overvakast. Dette gjeld fysisk kabelkonnektivitet så vel som optisk krysskoplingskonnektivitet på multipleks- og kanallaget. Tap av konnektivitet skal meddelast klientlaget.
- *Signalkvaliteten* skal overvakast på einskildkanalar så vel som på multipleksa kanalar.
- *Tilpassingsprosessane* mellom klientlaget og det tredelte tenarlaget, samt tenarlaga imellom, skal overvakast. Mellom klient og tenar gjeld dette prosessar som skal sikra ulike typar nyttelast frå klientane ei korrekt tilpassing ved oppkopling. Mellom tenarlaga gjeld dette prosessar for til dømes multipleksing/demultipleksing, allokering av optisk berebølgjelengde (frekvens), modulasjon/demodulasjon og gjenvinning av transporteining.
- *Indikasjon på defektar på dei einskilde tilknytingane i eit samband* skal rapporterast. Dette gjeld sekvensar innan alle dei tre laga.
- *Overlevingsevna* skal styrast. Dette dreier seg om systemet si evne til å gjenoppretta samband i ein feilsituasjon, til dømes ved å dynamisk kunna bruka ledig kapasitet. Det dreier seg òg om evna til å handtera reservevegar på meir statisk basis; etter 1:1 eller n:m-prinsippet.

Ein ser at desse krava vil medføra at drift- og styringssystema lyt ha tilgang til mykje lokal informasjon og kunna prosessera denne kontinuerleg. Det er mange og gode grunnar for at denne prosesseringa ikkje bør skje sentralisert, men distribuert.

Medan infrastrukturen for OTN har vorte standardisert i ITU-T, så er dette ikkje tilfelle for såkalla *control plane*-løysingar innan slike nett. Omgrepet *control plane* kjem opprinneleg frå svitsja telefonnett og omfattar først og fremst funksjonalitet knytt til signalering. Historisk har oppsett og nedtak av tilknytingar i dei svitsja tenestene vore kalla *svitsjing* og vore handtert av *control plane*. Dei samanliknbare funksjonane for linjer i transportnettet, har vore kalla *krysskopling* og vore handtert av *drift- og styringssystem*. Teknologisk er dette no eit kunstig og meiningslaust skilje, som skriv seg frå tida der ein delte denne funksjonalitet inn *svitsjing* og *transmisjon*.

Innanfor optiske nett er omgrepet *svitsjing* nytta synonymt med omgrepet *krysskopling*. Svitsjing er krysskopling mellom termineringspunkta for dei ulike tilknytingane som er skildra over. Svitsjing i optiske nett vert såleis realisert ved hjelp av krysskoplarar og multipleksarar. Dei tidlegare skilte fagfelta *svitsjing* og *transmisjon* vert såleis fullstendig integrert i det svitsja optiske transportnettet. Denne integrasjonen rekna ein med alt ved innføringa av *Synchronous Digital Hierarchy* (SDH) for 10-12 år sidan. SDH var eit konsept for *svitsja transportnett*, men har i hovudsak vorte realisert som eit *fastlinje-nett*. Dei opprinnelege krava til drift- og styring av SDH speglar svitsjeaspektet.

Det er dermed naturleg å sjå automatisert krysskopling (svitsjing) som ein automatisert konfigureringsfunksjon i eit *distribuert* drift- og styringssystem. Svitsjinga kan sjåast som automatisk krysskopling basert på drift- og styringsinformasjon og signalerings-/rutingsalgoritmar innanfor systemet. Svitsjinga vert dermed eit døme på ein distribuert prosess i slike system. Fleire store prosjekt har hatt dette perspektivet.

I den funksjonelle ITU-T-arkitekturen for transportnett vert grunnleggjande transportfunksjonar skildra utan nokon eigentleg referanse til *control* eller *management* av desse. Ein snakkar om *static* og *dynamic configuration management* for oppsett og nedtak av ulike typar tilkoplingar. TINA skiljer, som før nemnt, ikkje mellom *control* og *management* i modelleringa av nettressursar.

Å kunna handtera *control plane* i eit optisk nett krev (53):
- Ein veldefinert namne- og adresseplan.
- Ein rutingsprosess for å handtera topologi, ressursbruk og rutekalkulering.
- Eit signaleringsnett for kommunikasjon mellom entitetar som etterspør tenester og entitetar som leverer desse tenestene.
- Ein signaleringsprotokoll for oppsett, vedlikehald og nedtak av tilknytingar.

Førebels er tilknyttingane i OTN i all hovudsak faste linjer. Oppsett/nedtak er styrt via dei tradisjonelle og generelle drift- og styringsprotokollane. Sidan desse er utvikla for hierarkiske sentraliserte drift- og styringsstrukturar, vil dei truleg ikkje vera dei mest effektive til å handtera konnektivitet over horisontale grensesnitt mellom ulike domene. Protokollar utvikla spesielt for signaleringsformål vil nok vera langt meir effektive.

Kor vidt *control plane*-funksjonalitet/teknologi vert integrert i *configuration management*-funksjonalitet i det framtidige OTN er vanskeleg å spå om. Uansett er det vanskeleg å sjå tekniske fordeler ved eit skarpt skilje, særleg med omsyn til informasjonsbasar og naudsynt interaksjon mellom *control plane*-funksjonar og *andre* typar drift- og styrings-funksjonar som til dømes den generelle overvakinga/rapporteringa av konnektivitet og tilstand elles i nettet. Sterke kommersielle interesser innan dette problemkomplekset vil sjølvsagt òg vera drivarar for denne utviklinga. Optiske transportnett vil uansett kreva omfattande drift- og styringssystem.

Fleire store internasjonale prosjekt har sett på problemstillingar rundt drift og styring av OTN. Det DARPA-støtta amerikanske prosjektet MONET tek utgangspunkt i TMN/TINA i "*control and management*"-modellen sin (54). Dei har utvikla ein CORBA-basert prototype for eit distribuert drift- og styringssystem for optisk svitsja WDM-nett. Systemet opererer både på element- og nettlaget. Informasjonsmodellen er basert på standard TMN-objekt som er instansiert som CORBA-objekt. Informasjonsbasen skal vera tilgjengeleg for alle funksjonar. Systemet omfattar mellom anna algoritmar for innhenting av topologi- og statusinformasjon, og algoritmar som mogleggjer automatisk ruting (dynamisk val av rute, val av bølgjelengde og oppsett av sti (krysskopling) gjennom nettet). Agentbaserte signaleringsprotokollar er mellom anna nytta for desse formåla. Sjølv om signalerings-mekanismane støttar klientbasert handtering av tilkoplingar, fokuserte prosjektet i liten grad interoperabilitet mellom IP og WDM-laga.

Nokre større europeiske prosjekt har òg laga framlegg til informasjonsmodellar for drift- og styringsformål med utgangspunkt i TMN (54).

(55) foreslår ein ny objektklasse i TMN informasjonsmodell for å handtera automatisk optisk krysskopling. Ei rutingsmatrise som som er ei spesialisering av krysskoplingsfragmentet i TMN informasjonsmodell, er grunnlag for ein funksjonell modell som alternativt kan sjåast som funksjonell modell for *control plane*.

Dei viktigaste aktørane innanfor standardisering av ulike aspekt innan optiske transportnett er:
- ITU-T har fleire arbeidgrupper. Den eine ser mellom anna på problematikken rundt *control plane* og *distributed connection management*.
- IETF har hatt eit meir "IP-sentrisk" utgangspunkt ved å lansera rutings- og signaleringsprotokollar frå "IP-verda" for gjenbruk i optisk *control plane*. Særleg ser MPλS ut til å kunna nyttast for OTN. Sjå (49) for nærmare gjennomgang av dette. Utgangspunktet til IETF har mellom anna vore å raskt få fram ei løysing som kan handtera fleire ulike klientdata, og i første omgang ser det ut til MPλS vil verta nytta som svitsjeteknologi. IETF har fleire arbeidsgrupper for dette feltet.
- *Optical Internetworking Forum* (OIF) er òg sentrale i utviklinga av signalerings- og rutingsprotokollar for optiske nett.

6.2 Integrert drift og styring

I tillegg til *distribusjon* er *integrasjon* eit sentralt omgrep innan utvikling av framtidige drift- og styringssystem. Ein vil i dette avsnittet gå gjennom ulike felt der integrasjon er aktuelt.

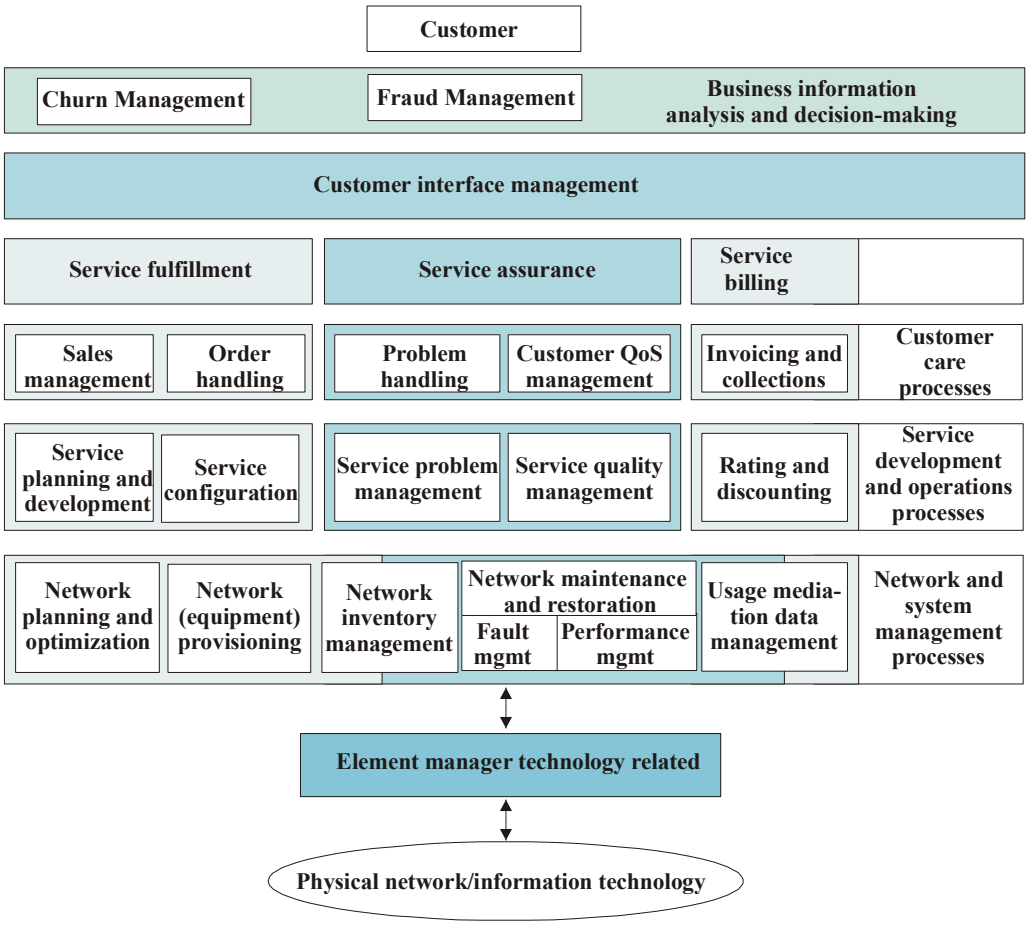
6.2.1 Integrasjon i drift og styring av nett og tenester

Som før nemnt ynskjer ein ved hjelp av mellomvareteknologi å laga ei plattform for utvikling av tenester. Dette kan i utgangspunktet vera applikasjonar for alle typar tenester som skal tilbydast i eller via eit kommunikasjonensnett; til dømes post, underhaldning, handel. Ein skal i det følgjande kort gå gjennom dei mest sentrale områda for utvikling av tenestelaget for drift- og styring, dvs tenester som gjeld sjølve kommunikasjonsnettet. Applikasjonane som vil koma på dette området, vil vera avhengige av eit samspel med eksisterande applikasjonar for drift og styring av element- og nettlaget. Desse tenesteapplikasjonane vil i stor grad baserast på ei aggregering/abstrahering av informasjon frå laga under. Ei viktig årsak til at fokus no ser ut til å flyttast frå drift og styring av *element/nett* til drift og styring av *tenester*, er at aktørar utan stor teknisk kompetanse (kundar, tenestetilbydarar) i større grad vil krevja tilgang til styrings- og kontrollfunksjonar i kommunikasjonsnettet. Dette må medføra enklare system og tenester. Dei viktigaste funksjonane på tenestelaget ser ut til å vera:
- *Handtering av tenestekvalitet*. Dette er ein viktig funksjon som er naudsynt for å best mogleg kunna garantera yteevne uavhengig av underliggjande transportteknologi. Gode system for dette formålet må kunna handtera informasjon frå fleire nett.
- *Ordrehandtering*. Dette er ein funksjon for å kunna administrera ordrar frå kundar; etablera ein arbeidsflyt og distribuera ifølgje denne.
- *Problemhandtering*. Dette er ein funksjon for å kunna administrera problem ved dei ulike tenestene. Dette vil først og fremst vera problem meldt av kunde. Funksjonen må i tilleg til å laga problemrapportar, omfatta monitorering status, prioritet og eskaleringsnivå.
- *Kundetenestehandtering*. Dette er ein funksjon for å kunna handtera interaksjon mellom kunde og leverandør.
- *Service Level Agreement (SLA)-handtering*. SLA mellom to partar spesifiserer eit avtalt nivå for til dømes yteevne og konnektivitet for ei gitt teneste. Oppfølgjing av SLA vil vera avhengig av funksjonalitet knytt til drift og styring av nettelement og nett. I tillegg kjem handtering av kundespesifikke terskelverdiar, handtering av misleghald av kontrakten samt jevnleg rapportering til kunden.
- *Fakturering*. Dette er òg funksjonalitet som i stor grad er basert på måledata frå ulike

element i ulike nett. Vern mot svindel vil òg få fokus.

- *Regelbasert (policy-based) drift og styring*. Regelstyring vil verta nytta i tilfelle der det krevs komplisert analyse og rask reaksjon. Dette gjeld til dømes i handteringa av SLA. Ein *regel* er i denne samanhengen ein formell representasjon av informasjon vedrørande nettet/elementet, men er spesifisert uavhengig av desse komponentane. Slik funksjonalitet er særleg aktuell for å kunna styra tenestekvalitet, datasikring og inter-domene-konfigurasjon.
- *Interdomene-funksjonar*. Dette er eit sett med funksjonar for å kunna koordinera drift og styring av nett på tvers av domene. Inter-domenefunksjonalitet skal handtera alle dei fem funksjonelle områda for drift og styring av nett. Funksjonane skal mellom anna sikra kunden *eitt* kontaktpunkt uavhengig av kor mange domene som er involvert i ein feilsituasjon, eller i oppsett av eit samband/oppkopling/teneste.

Store operatørar har sjølvsagt ei rekkje system for dei fleste av desse funksjonane. Systema kan vera meir eller mindre integrert med trafikknettet og/eller med eksisterande drift- og styringsystem for nett. Fleire av dei nemnte tenestelagsfunksjonane er standardisert for TMN. Utfordringa ligg i integrasjon med drift- og styringssystema for element-/nettlaget, men kanskje først og fremst i interdomenefunksjonaliteten andsynes andre operatørar, mange og ulike tenestetilbydarar samt kundar.



Figur 6.1 Telecom Operation Map (56)

Figur 6.1 syner *Telecom Operation Map* (TOM), som ser ut til å få stor aksept. Modellen er utarbeidd av *Tele Management Forum* (TMF) og prøver å syna ei avbilding mellom drift- og

styringsfunksjonar for nett- og tilsvarande funksjonar for tenestelaget. Vertikalt deler ein prosessane inn etter kva som må til for at tenesta skal kunna innfriast, kvalitetssikrast og fakturerast. Horisontalt deler ein prosessane inn i kundehandtering, tenesteutvikling og operative prosessar samt drift og styring av nett og system.

6.2.2 Integrasjon i drift og styring av ulike nettteknologiar

Det er ikkje berre på tenestelaget ein ynskjer å kunna operera "på tvers" av ulike teknologiar. Operatørar ynskjer å unngå ein situasjon der einkvar introduksjon av ny teknologi i nettet medfører eit nytt spekter av IT-system som understøttar dei ulike prosessane for den eine teknologien åleine. Ein ynskjer å integrera drift- og styringsfunksjonalitet for fleire teknologiar. Dette gjeld til dømes funksjonalitet for *feilhandtering*. Ein ynskjer å kunna "sjå" avvik og feilsituasjonar på tvers av teknologiar og å kunna korrelera alarmar frå ulike utstyrtypar. Dette vil til dømes vera avgjerande i samband med proaktiv feilhandtering. Eit anna aktuelt felt for slik integrering er funksjonalitet for *bruksregistrering* for å betra takserings- og faktureringsprosessane.

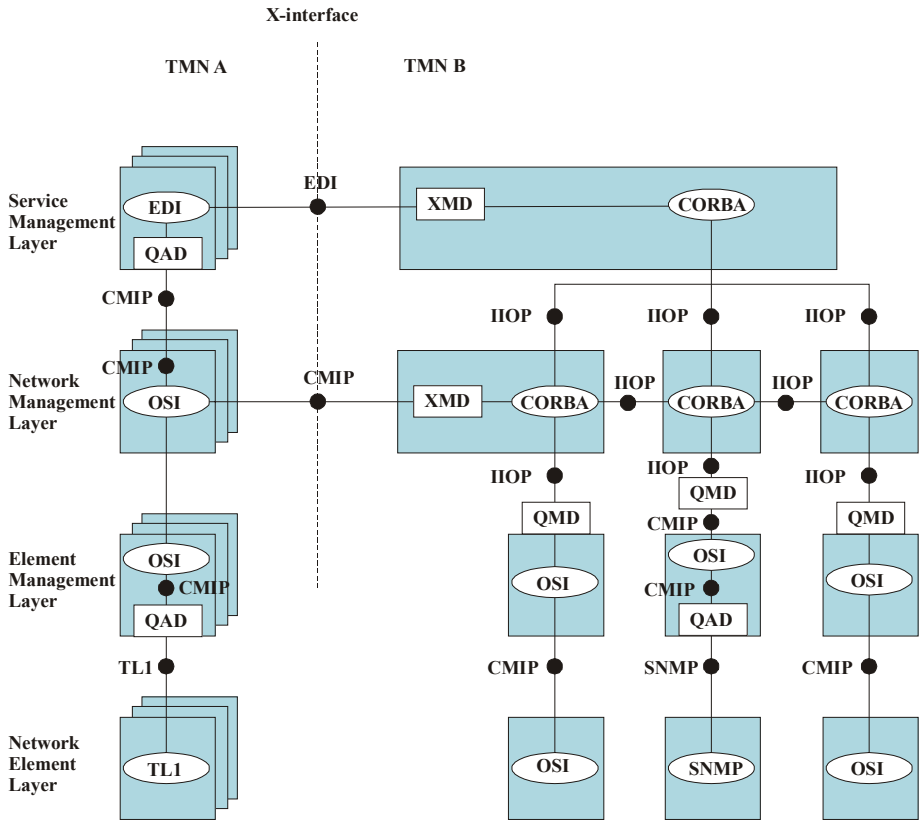
Eit anna problem her er at ulike leverandørar utviklar spesifikke løysingar innan ein og same teknologi. Dette gjeld i første rekkje funksjonalitet for *konfigurasjon*. Dette har vore eit stort problem innan konfigurering av telefonsentralar; like eins i konfigurering av SDH-teknologi.

6.2.3 Integrasjon av ulike drift- og styringsteknologiar

Eit viktig aspekt ved dei fleste arkitekturane som vart gjennomgått i kapittel 5, er at dei søkjer å integrera SNMP- og OSI/TMN-basert drift og styring. I snart eit tiår har dette vore tema for ulike arbeidsgrupper:

- *ISO/CCITT and Internet Management Coexistence* (IIMC) har arbeidd med spesifikasjonar for oversetting mellom informasjonsstruturane GDMO og SMI, så vel som mellom protokollane CMIP og SNMP.
- *Joint Inter-Domain Management* (JIDM) har som mål å laga gateways mellom CORBA-baserte applikasjonar på den eine sida og OSI- eller SNMP på den andre. Ei slik løysing vil ikkje vera avhengig av endringar i SNMP/OSI-agentane.

Beggje desse gruppene overset drift- og styringsinformasjonen på *syntaks*nivå. DMTF ynskjer å etablera CIM som ein framtidig informasjonsmodell, sjå avsnitt 5.6.2. Dersom dei lukkast med dette og greier å avbilda dei mange Internet MIBs og tilmed OSI MIBs til CIM på *semantisk* nivå, kan dette mogleggjera ein langt djupare integrasjon av drift- og styringssystem.



Figur 6.2 Døme på integrasjon av ulike drift- og styringsteknologiar (19)

Figur 6.2 syner korleis ein tenkjer seg integrasjon innanfor og mellom TMN-domene. Informasjons- og protokollkonvertering vil gå føre seg i meklings- og tilpassingsgrensesnitta, sjå avsnitt 3.4.1. Å avbilda TMN til CORBA utan å missa funksjonalitet, har synt seg å ikkje vera trivielt (57).

Sameksistens mellom dei ulike SNMP-versjonane er skildra i (41).

6.3 Andre teknologiar

I tråd med at funksjonaliteten i netta vert distribuert, vil drift- og styringsfunksjonar verta flytt ned mot nettelementa for å gjera desse meir ”sjølvstyrte”. Ein tenkjer seg at netta i framtida i stor grad vil vera ”sjølvregulerande” med omsyn til trafikklast og overlevingsevne. I denne samanhengen vil proaktiv problem- og feilhandtering vera viktig. Drift- og styringssystem for slike autonome nett vil til dømes ha komponentar for ”nett-diagnose” og ”læring”, for til dømes å kunna justera trafikken i takt med endringar i nettet.

Den objektorienterte mellomvareteknologien som er omtalt tidlegare i denne rapporten er eit viktig grunnlag for vidare utvikling fram mot autonome nett. Ein skal i resten av dette avsnittet kort presentera nokre andre teknologiar som vil stå sentralt.

6.3.1 Agentteknologi

Det kan vera uklart kva omgrepet *agentteknologi* omfattar. Ein sentral eigenskap ved eit

agentsystem er at det er *autonomt* (58). Ein kan her sjå ein analogi til *objektet* i eit objektorientert system. Objektet innkapslar ein tilstand. Objektet har kontroll over tilstanden i den forstand at tilstanden kan endrast berre ved hjelp av metodar objektet sjølv leverer. Agenten kapslar på same vis inn ein tilstand, men kapslar i motsetning til objektet òg inn *åtferd*. Det medfører at agenten, i motsetning til objektet, òg har kontroll over åtferda, dvs eigne aksjonar. Eit objekt *a* kan *påkalla* ein metode frå objekt *b*. Objekt *b* har ingen kontroll over dette. Objektet er ikkje autonomt. Ein agent *a* kan ikkje påkalla noko frå agent b, men må *be* om å få ein aksjon utført. Agent *b* avgjer om aksjonen skal utførast. Agenten er autonom. Med omgrepet *agentbasert system* meinast eit system der nøkkelabstraksjonen er *agenten*.

Det finns ei rekkje, både enkle og svært komplekse, autonome sanntidssystem for til dømes prosesstyring. Det finns òg autonome såkalla *deamons* for software-monitorering. Desse vert vanlegvis ikkje kategorisert som agentteknologi. Eit agentsystemet må i tillegg vera i stand til å utføra *fleksible* autonome aksjonar. Agenten må kunna vera:

- *Lydhøyr* overfor omgjevnadene og kunna reagera på endringar i tide. Omgjevnad kan i denne samanhengen vera eit fysisk element, ein brukar eller eit nett.
- *Proaktiv* ved ikkje berre å kunna reagera på endringar, men ógså å kunna oppvisa målretta åtferd ved å ta initiativ til aksjon.
- *Sosial* ved å vera i stand til problemløysing ved interaksjon med andre agentar og menneske.

Agentsystem med slike eigenskapar vert av mange kalla *intelligente agentar*. Ei rekkje termar er nytta for å skildra/vektleggja ulike eigenskapar ved slike applikasjonar: Mobile agentar, informasjonsagentar, personlege agentar, lærande agentar, samarbeidande agentar osv.

Området er i rask utvikling. Både AgentX og SNMPv3, som vart gjennomgått i kapittel 4, har vorte utvikla som modulære agentstrukturar. Den omfattande bruken av web-lesarar, web-applikasjonar og Java har framskaffa ein infrastruktur for å kunna utvikla og implementera ny agentteknologi, sjå avsnitt 5.6.1.

I samband med drift- og styringssystem vil bruk av agentteknologi seia at ein basisagent er til stades i systemet som skal styrast. Drift- og styringsfunksjonalitet vert konfigurert på, eller lasta ned til, denne agenten etterkvart som nye funksjonar trengs. Ein drift- og styringsagent er hovudsakleg sett saman av følgjande komponentar (1):

- *Agent-kjerne*. Denne styrer dei andre komponentane ved å mogleggjera kommunikasjon mellom dei, distribuera innkomande forespørslar, tillata managerapplikasjonar å installera nye eller sletta gamle komponentar.
- *Basis drift- og styringstenester*. Dette kan til dømes vera notifikasjonstenester eller filtrering.
- *Protokolladapter* som mogleggjer kommunikasjon over SNMP, CMIP, IIOP, RMI, HTTP osv.
- *Drift- og styringskomponentar* som utfører spesifikke oppgåver. Desse vert typisk lasta ned til agentkjernen når funksjonane trengs. Komponenten kan bruka basistenestene og protokolladapter.

Denne teknologien er sentral når ein ynskjer å automatisera og desentralisera drift- og styringfunksjonalitet. Prinsippet er at funksjonalitet *dynamisk* kan delegerast til nettelementa og utførast lokalt. I staden for å flytta data frå element til applikasjon, flytter ein såleis applikasjonen til elementa der dataene er. Dette prinsippet kallast gjerne *Management by*

Delegation (MbD).

Metoden er eit alternativ til tradisjonelle konsept for objektorientert distribuert prosessering, der ein flytter data til og frå distribuerte applikasjonar. Ved hjelp av agentteknologi ynskjer ein å kunna flytta drift- og styringsapplikasjonar til distribuerte datakjelder.

Kritikarar trekkjer gjerne fram problem knytt til interoperabilitet (59). Dette gjeld mellom anna ein generell mangel på standardar, mangel på domenespesifikke ontologiar og mangel på ein berande agentstruktur for MbD, slik ein har i til dømes CORBA for distribuerte objektorienterte system.

6.3.2 Aktive Nett

Aktive nett, eller *progarmmerbare nett*, er eit relativt nytt konsept. I motsetning til tradisjonelle nett, som hovudsakleg leverer transportmekanismar for å overføra data frå eit endesystem til eit anna med eit minimum av dataprosessering, så vil aktive nett:

- Tillata nettnodane å handsama sendte data opp til applikasjonslaget.
- Tillata ein brukar å senda sin eigen programkode til nettnodane for at desse skal kunna handsama data brukaren sender.

Dataprosessering innan transportnodar som rutarar og svitsjar, er i dag avgrensa til funksjonalitet opp til lag tre i OSI-modellen. Prosesseringa er knytt til signalering, ruting, trafikkflytkontroll osv. Slike passive nett har ibuande problem ved at det er vanskeleg å integrera ny teknologi, nye standardar og nye tenester når det tekniske miljøet er heterogent. Ytinga vert dessutan sett ned på grunn av redundante operasjonar på dei ulike protokollaga. Siste tida har det dessutan vorte utvikla applikasjonar som krev brukarstyrt dataprosessering i nettnodane, og ein har laga ulike ad hoc-løysingar som i utgangspunktet er uønska. For å løysa desse problema ynskjer ein å utvikla ein arkitektur med generisk evne til å handtera brukarstyrt programmering av nettet. I eit slikt nett vil mellomliggjande transportnodar kunna operera på kommunikasjonslag sju; applikasjonslaget.

Likskapen mellom agent-teknologi og aktive nett er stor, og mange av prosjekta som forskar på aktive nett, nyttar agent-teknologi (60). Ideen bak aktive nett er likevel langt meir generell og omfattande, ved at ein ser føre seg eit kommunikasjonsnett som ei samling av:

- Aktive nodar som kan utføra generell datahandsaming.
- Aktive pakkar som inneheld programkode og som faktisk *er* program.

I eit slikt perspektiv vil mobile agentar vera spesifikke typar av aktive pakkar.

Kritikarane av konseptet har mellom anna stillt desse spørsmåla:

- Vil aktive nett medføra auka yteevne for applikasjonane (kost/nytte)?
- Vil kompleksiteten i aktive nett undergrava suksessen til Internet, - ein suksess som nettopp er basert på det enkle?
- Vil den omfattande datasikringsproblematikken, som aktive nett heilt klart medfører, vera handterbar?

Det har hittil vore to ulike innfallsvinklar for arkitekturar for slike nett:

- *Aktiv pakke-tilnærming*. Denne måten er karakterisert ved at koden som noden skal utføra

vert overført i aktive pakkar. Noden har i utgangspunktet ingen aktiv kode lagra. Noden er aktiv ved at han tillet prosessering opp til applikasjonsnivå. Programmet som vert overført i pakken skal utførast, enten på data som er overført i same pakke, eller for å endra tilstanden eller åtferda til noden sjølv.

- *Aktiv node-tilnærming*. Denne måten er karakterisert ved at koden som noden skal utføra, på førehand vert lagra i den aktive noden. Pakken har ingen kode men inneheld referansar til predefinerte funksjonar i noden. Pakken er aktiv ved at han avgjer kva funksjonar som skal utførast på dataene og leverer parametrar til desse.

Då det er fordeler og ulemper ved beggje desse metodane, prøver fleire prosjekt no å sameina dei i eitt og same system.

Active Network Group arbeider med ein ny protokoll, *Active Network Encapsulation Protocol* (ANEP). Andre foreslår ei utviding av IP; *Active IP*.

Eit av områda som alt har peikt seg ut som særleg relevant for denne teknologien er drift og styring av kommunikasjonsnett. Motivasjonen er langt på veg den same som for tidlegare omtalte konsept basert på objektorientert eller agentbasert distribuert prosessering:

- Avvik, problem og feil i nettet vil kunna avdekkast og rapporterast raskt og automatisk.
- Bandbreidda i nettet vert betre utnytta, til dømes ved at ein unngår *polling* og redundant informasjon
- Sidan nodane i utgangspunktet er utstyrt med ein viss ”intelligens”, kan mange avgjersler takast der problema oppstår. Eit operativt senter kan dessutan senda høveleg programkode til noden for at denne skal kunna reparera seg sjølv.

Aktive nett vil opplagt gjera multi-domene-problematikken meir kompleks. På den andre sida kan aktive nett ha potensiale til nettopp å kunna løysa desse problema, sidan nye reglar og standardar raskt kan etablerast i tråd med den ibuande fleksibiliteten programmerbare nodar vil ha.

Mykje forsking og utvikling står att innan denne teknologien, som førebels ikkje er prøvd i storskala nett. Dersom ein lukkast med dette konseptet, vil drift- og styringssystema få nye utfordringar, samstundes som dei sjølve vil vera basert på aktiv-nett-teknologi. Datasikring ser i dag ut til å verta den viktigaste utfordringa. (61) gir ein oversikt over trugsmål og moglege mottiltak. Sidan aktive nett er langt meir fleksible enn passive, vil dei vera langt meir sårbare for mistak og utilsikta feiloperasjon så vel som for direkte åtak. Talet på moglege hòl i tenester og mekanismar for data- og nettsikring vil auka dramatisk. Ein aktiv pakke kan i utgangspunktet øydeleggja (deler av) nettet ved å endra tilstandar og prosessar, sletta data, generera uhandterleg store trafikkmengder, øydeleggja andre pakkar osv. Då passive og aktive komponentar vil eksistera side om side, vil den nye trugsmålsmodellen ikkje erstatta den gamle, men koma som tillegg til denne. Fleire prosjekt ser på datasikring i aktive nett, men førebels er dette eit såpass nytt felt at det er utråd å seia kva løysingar som vil verta nytta.

Sjølv om problema andsynes drift og styring generelt og datasikring spesielt kan synast uoverstigelege, peikar mykje i retning av at konseptet har eit potensiale som gjer at det er grunn til å rekna det som ein farbar veg for framtidige kommunikasjonsnett. Aktive nett ser derimot ikkje ut til å vera rett rundt hjørnet. Ein overgang til aktive nett vil etter mykje å døma skje

gradvis og etter kontinuerleg avveging mellom fleksibilitet og kompleksitet samt mellom sikringsnivå og yteevne.

6.4 Oppsummering

Nokre klare trendar i utviklinga av drift- og styringskonsept er gjennomgått. Trendane på dette feltet følgjer arkitekturtrendane for kommunikasjonsnett generelt, og dei kan samanfattast slik:

- Distribuerte system som mellom anna skal mogleggjera raskare og meir proaktiv handtering av nettet. Problemstillingar innan drift og styring av heil-optiske nett, syner at gamle grenser mellom svitsjestyring og transmisjonsstyring no er uklare, om dei i det heile finns.
- Integrert drift og styring av nett og tenester, integrerte system som handterer ulike netteknologiar og interoperable system som greier å sameina ulike drift- og styringsteknologiar.
- Objektorientert teknologi som mellom anna mogleggjer ein abstraksjon av kompleksiteten i nettet og legg grunnlag for rask utvikling av tenester/applikasjonar og effektiv drift og styring av desse.
- Agentteknologi som mellom anna skal mogleggjera større grad av sjølvregulering og sjølvstyring i nettet.
- Aktive nett som representerer eit heilt nytt konsept for kommunikasjonsnett. Utfordringane andsynes drift og styring av slike nett er enorme. Samstundes kan det sjå ut til at denne teknologien nettopp vil vera den som vil kunna handtera dei store problema ein alt i dag ser med omsyn til heterogene nett og tenester, samt eit raskt aukande tal på aktørar med ulike krav og målsettingar.
- Datasikring vil truleg verta ei av dei største utfordringane for framtidige drift- og styringssystem.

7 OPPSUMMERING

Ein har i denne rapporten gått gjennom dei viktigaste standardane og arkitekturane for drift og styring av kommunikasjonsnett.

Dei dominerande arkitekturane i dag er:
- TMN som fullt ut er basert på OSI-standardane, men som òg representerer ei omfattande konkretisering og utviding av desse. TMN dominerer i dei tradisjonelle telenetta.
- System basert på SNMP-standardane. Desse dominerer i lokalnett og i IP-nett generelt.

Standardane er sjølvsagt prega av dei krava som har vore stillt til kommunikasjonsnetta dei skulle handtera. TMN skulle handtera nett som tradisjonelt har hatt store og definerte krav til yting, tenestekvalitet, feilhandtering og datasikring. Spesifikasjonane er omfattande og til dels komplekse, både med omsyn til funksjonalitet, informasjonsrepresentasjon og kommunikasjon. SNMP skulle handtera nett der krava tradisjonelt meir har vore knytt til raske endringar i nettstruktur og konfigurasjon. Ein har så langt råd prøvd å gjera informasjonsmodellen så vel som protokollen, så enkel som råd. Ei årsak til dette er sjølvsagt også at desse standardane var meint å skulle vara berre ei kort tid.

Sjølv om standardane er svært ulike, representerer beggje desse arkitekturane eit tradisjonelt og sentralisert drift- og styringskonsept ved at hendingar i nettet vert innrapportert til ein sentral applikasjon. Standardar og spesifikasjonar er gjennomgått med omsyn til funksjonalitet, informasjonsstruktur, informasjonsinnhald, kommunikasjonsmodell og datasikring.

Nyare arkitekturar er presentert. Desse er i langt større grad enn dei to tradisjonelle arkitekturane farga av konvergensen mellom tradisjonelt skilte tele- og datanett, og er fundert på at moderne kommunikasjonsnett er distribuerte datasystem. Dei nye arkitekturane representerer eit distribuert konsept ved at hendingar i nettet i stor grad skal handterast av lokale applikasjonar. Ein ynskjer vidare at systema skal kunna forebyggja feil og problem i nettet.

Dei nyare arkitekturane for kommunikasjonsnett generelt, legg stor vekt på å gje ei plattform for rask utvikling av tenester og applikasjonar ved å skjula kompleksiteten i det underliggjande nettet. Dette vert spegla i konsepta for drift og styring. Drift og styring av dei distribuerte tenestene/applikasjonane representerer på mange vis ein ny dimensjon i høve til noverande drift- og styringssystem.

Til slutt er dei klaraste utviklingstrendane innan drift- og styringskonsept gjennomgått. Desse skal fungera i sjølvregulerande og programmerbare kommunikasjonsnett og vil møta store utfordringar mellom anna med omsyn til datasikring.

APPENDIX

A DEFINISJONAR

Dette vedlegget inneheld definisjonar frå ITU-T-rekommendasjonar og IETF-spesifikasjonar. Det syner vidare korleis termane er omsett til norsk i denne rapporten. Nokre av termane på denne lista er ikkje nytta i rapporten, men er med på denne lista fordi dei inngår i definisjonen av andre termar.

Det finns døme på at ITU-T og IETF definerer ein og same term ulikt. I rapportteksten vil det gå fram av samanhengen om det er ITU-T- eller IETF-definisjonen som gjeld.

A.1 Definisjonar frå ITU-T-rekommendasjonar

Term frå ITU-T-rekommendasjonar	**Definisjon frå ITU-T-rekommendasjonar**	**Term nytta i denne rapporten**
(N)-association	A cooperative relationship among (N)-entity-invocations.	(N)-assosiasjon
(N)-connection	An association requested by an (N+1)-entity for the transfer of data betweeen two or more (N+1)-entities. The association is established by the (N)-layer and provides explicit identification of a set of (N)-data-transmissions and agreement concerning the (N)-data-transmission services to be provided for the set.	(N)-tilknyting
(N)-data-transmission	An (N)-facility which conveys (N)-service-data-units from one (N+1)-entity to one or more (N+1)-entities.	(N)-data-transmisjon
(N)-entity	An active element within an (N)-subsystem embodying a set of capabilities defined for the (N)-layer that corresponds to a specific (N)-entity-type (without any extra capabilities being used).	(N)-entitet
(N)-entity-type	A description of a class of (N)-entities in terms of a set of capabilities defined for the (N)-layer.	(N)-enititetstype
(N)-facility	A part of an (N)-service.	(N)-fasilitet
(N)-function	A part of the activity of (N)-entities.	(N)-funksjon
(N)-layer	A subdivision of the OSI architecture, constituted by subsystems of the same rank (N).	(N)-lag
(N)-protocol	A set of rules and formats (semantic and syntactic) which determines the communication behavior of (N)-entities in the performance of (N)-functions.	(N)-protokoll
(N)-protocol-control information	Information exchanged between (N)-entities to co-ordinate their joint operation.	(N)-protokoll styringsinformasjon
(N)-protocol-data-unit	A unit of data specified in an (N)-protocol and consisting of (N)-protocol-control-information and possibly (N)-user-data.	(N)-PDU, (N)-protokoll dataeining
(N)-service	A capability of the (N)-layer and the layers beneath it, which is provided to (N+1)-entities at the boundary between the (N)-layer and the (N+1)-layer.	(N)-teneste
(N)-service-data-unit	An amount of information whose identity is preserved when transferred between peer-(N+1)-entities and which is not interpreted by the supporting (N)-entities.	(N)-SDU, (N)-teneste dataenhet
(N)-subsystem	An element in a hierarchical division of an open system which interacts directly only with elements in the next higher division or the next lower division of that open system.	(N)-subsystem
(N)-user-data	The data transferred between (N)-entities on behalf of the (N+1)-entities for whom the (N)-entities are providing services.	(N)-brukardata
(systems) managed	A managed object relevant to more than one layer, to the system as a	styrt (system-)

object	whole, or to specific management functions.	objekt
(systems) management	Functions in the Application Layer related to the management of various OSI resources and their status across all layers of the OSI architecture.	drift og styring av system
(systems) management application process	An application process participating in systems management	applikasjons-prosess for drift og styring av system
(systems) management application protocol	An application layer protocol supporting systems management services	applikasjons-protokoll for drift og styring av system
(systems) management application service element	An application service element providing systems management services	applikasjonsteneste for drift og styring av system
(systems) management application-entity	An application-entity for the purposes of systems-management communications.	applikasjonsentitet for drift og styring av system
(systems) management function	A part of systems management activities which satisfy a set of logically related user requirements.	funksjon for drift og styring av system
(systems) management functional area	A category of systems management user requirements	funksjonsområde for drift og styring av system
access control	The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner.	tilgangskontroll
accountability	The property that ensures that the actions of an entity may be traced uniquely to the entity.	bruksregistrerbar sporbar
action	An operation on a managed object, the semantics of which are defined as part of the managed object class definition.	aksjon
active threat	The threat of a deliberate unauthorized change to the state of the system.	aktivt trugsmål
agent	An MIS-User, which for a particular systems management interaction, has taken an agent role.	agent
agent role	A role taken by an MIS-User in which it is capable of performing management operations on managed objects and of emitting notifications on behalf of managed objects.	agentrolle
allomorphism	The ability of a managed object that is an instance of a given class to be managed as an instance of one or more other managed object classes.	allomorfi
application process	An element within a real open system which performs the information processing for a particular application.	applikasjonsprosess
application-entity	An active element, within an application process, embodying a set of capabilities which is pertinent to OSI and which is defined for the Application Layer, that corresponds to a specific application-entity-type (without any extra capabilities being used).	applikasjonsentitet
attribute group	A group of attributes which have been assigned a single identifier for ease of access.	attributt-gruppe
audit	See security audit	-
authentication	See data origin authentication, and peer entity authentication	autentisering
authentication information	Information used to establish the validity of a claimed identity.	autentiserings-informasjon
authorization	The granting of rights, which includes the granting of access based on access rights.	autorisasjon
availability	The property of being accessible and useable upon demand by an authorized entity.	tilgjenge
behaviour	The way in which managed objects, name bindings, attributes, notifications and actions interact with the actual resources they model and with each other.	åtferd
capability	A token used as an identifier for a resource such that possession of the token confers access rights for the resource.	-
channel	An information transfer path.	kanal
characteristic	An element of a managed object class definition; that is an attribute definition, an attribute group definition, a notification definition, a behaviour definition, a parameter definition or a package definition	karakteristikk
ciphertext	Data produced through the use of encipherment. The semantic content of the resulting data is not available.	siffer

	Note – Ciphertext may itself be input to encipherment, such that super-enciphered output is produced.	
cleartext	Intelligible data, the semantic content of which is available.	klartekst
conditional package	A package which is present in a given managed object if the condition given in its managed object class definition is satisfied	vilkårspakke
confidentiality	The property that information is not made available or disclosed to unauthorized individuals, entities, or processes.	konfidensialitet
containment	A structuring relationship for managed objects in which the existence of a managed object is dependent on the existence of a containing managed object.	-
cryptanalysis	The analysis of a cryptographic system and/or its inputs and outputs to derive confidential variables and/or sensitive data including cleartext.	kryptoanalyse
cryptographic checkvalue	Information which is derived by performing a cryptographic transformation (see cryptography) on the data unit. *Note* – The derivation of the checkvalue may be performed in one or more steps and is a result of a mathematical function of the key and a data unit. It is usually used to check the integrity of a data unit.	kryptografisk sjekkverdi
cryptography	The discipline which embodies principles, means, and methods for the transformation of data in order to hide its information content, prevent its undetected modification and/or prevent its unauthorized use. *Note* – Cryptography determines the methods used in encipherment and decipherment. An attack on a cryptographic principle, means, or method is cryptanalysis.	kryptografi
data integrity	The property that data has not been altered or destroyed in an unauthorized manner.	dataintegritet
data origin authentication	The corroboration that the source of data received is as claimed.	autentisering av datakjelde
denial of service	The prevention of authorized access to resources or the delaying of time-critical operations.	teneste-avslag
digital signature	Data appended to, or a cryptographic transformation (see cryptography) of a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery e.g. by the recipient.	digital signatur
encapsulation	A relation between a managed object and its attributes and behaviour, which represents the property that attributes and behaviour may be observed only through management operations on the managed object or notifications emitted by it.	innkapsling
encipherment	The cryptographic transformation of data (see cryptography) to produce ciphertext. *Note* – Encipherment may be irreversible, in which case the corresponding decipherment process cannot feasibly be performed.	-
encryption	See encipherment.	kryptering
inheritance	The conceptual mechanism by which attributes, notifications, operations and behaviour are acquired by a subclass from its superclass.	arv
inheritance hierarchy	A hierarchical arrangement of managed object classes where the hierarchy is organized on the basis of the class specialization.	arvehierarki
instantiation	The process of creating a managed object according to a managed object class definition	instansiering
integrity	See data integrity	integritet
key	A sequence of symbols that controls the operations of encipherment and decipherment.	nøkkel
key management	The generation, storage, distribution, deletion, archiving and application of keys in accordance with a security policy.	nøkkelhandtering
managed (open) system	A real open system containing an MIS-User which can take the agent role.	styrt (ope) system
managed object	The OSI management view of a resource that may be managed through the use of OSI management protocol(s).	styrt objekt
managed object class	A named set of managed objects sharing the same (named) sets of attributes, notifications, management operations (packages), and which share the same conditions for presence of those packages. NOTE – The following two definitions are aligned with the corresponding definitions in OSI Conformance Testing Methodology and Framework ITU-T Rec. X.290 | ISO/IEC 9646-1 for PICS and	styrt objektklasse

	PICS proforma.	
management domain	A specification of a collection of managed objects of interest.	drift- og styringsdomene
management information	The information within an open system which may be transferred by OSI management protocols.	drift-og styringsinfomasjon
management information base (MIB)	The conceptual repository of management information within an open system.	drift-og styrings-infomasjonsbase
manager	A Mangement Information Service (MIS)-User, which for a particular systems management interaction, has taken a manager role.	manager
manager role	A role taken by an MIS-User in which it is capable of issuing management operations and of receiving notifications.	managerrolle
managing (open) system	A real open system containing an MIS-User which can take the manager role	styrande system
mandatory package	A package which must be present in all instances of a given managed object class	obligatorisk pakke
Mangement Information Service (MIS)-User	An application making use of systems management services.	drift- og styrings-informasjonsteneste
multiple inheritance	A conceptual mechanism that allows a subclass to acquire attributes, notifications, operations and behaviour from more than one superclass.	multippel arv
name binding	A relation between object classes which specifies that an object of one identified class may be the superior of an object of another named class. A name binding definition also includes other information about the relation, and may be defined to also apply to subclasses of the superior or the subordinate class or both	namnebinding
naming schema	A collection of name bindings	namneplan
naming tree	A hierarchical arrangement of objects where the hierarchy is organized on the basis of the name binding relationship. An object used to name another managed object is higher in the hierarchy than the named object. The naming object is referred to as the superior of the named object, which is referred to as the subordinate	namnetre
notarization	The registration of data with a trusted third party that allows the later assurance of the accuracy of its characteristics such as content, origin, time and delivery.	bruk av notar
notification	Information emitted by a managed object relating to an event that has occurred within the managed object.	notifikasjon
notification type	A named data-type defining a specific kind of notification.	notifikasjons-type
open system	The representation within the Reference Model of those aspects of a real open system that are pertinent to OSI.	ope system
Open System Interconnection Environment (OSIE).	An abstract representation of the set of concepts, elements, functions, services, protocols, etc., as defined by the OSI Reference Model and the derived specific standards which, when applied, enable communications among open systems.	-
OSI management	The facilities to control, coordinate and monitor the resources which allow communications to take place in the OSI environment.	OSI drift og styring
OSI resource	Data processing and data communication resources which are of concern to OSI.	OSI-ressurs
OSI-(N)-Relay System	An open system which, for a particular instance of communication, makes use of OSI functions up to and including functions of the (N)-layer and where a relay function is executed within the (N)-layer	OSI-(N)-videresendings-system
package	A collection of attributes, notifications, operations and/or behaviour which is treated as a single module in the specification of a managed object class. Packages may be specified as being mandatory or conditional when referenced in a managed object class definition	pakke
parameter	A value of a type which has associated semantics and is associated with an object identifier and other information where the value of the type may be carried in protocol.	parameter
passive threat	The threat of unauthorized disclosure of information without changing the state of the system.	passivt trugsmål
peer-(N)-entities	Entities within the same (N)-layer.	lag-(N)-entitetar
peer-entity authentication	The corroboration that a peer entity in an association is the one claimed.	autentisering av lag-entiteter innen gitt lag

privacy	The right of individuals to control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed. *Note* – Because this term relates to the right of individuals, it cannot be very precise and its use should be avoided except as a motivation for requiring security.	(rett til) personvern
real open system	A real system which complies with the requirements of OSI standards in its communication with other real systems.	verkeleg (ope) system
real system	A set of one or more computers, the associated software, peripherals, terminals, human operators, physical processes, information transfer means, etc., that forms an autonomous whole capable of performing information processing and/or information transfer.	verkeleg system
repudiation	Denial by one of the entities involved in a communication of having participated in all or part of the communication.	fornekting
routing control	The application of rules during the process of routing so as to chose or avoid specific networks, links or relays.	rutingskontroll
rule-based security policy	A security policy based on global rules imposed for all users. These rules usually rely on a comparison of the sensitivity of the resources being accessed and the possession of corresponding attributes of users, a group of users, or entities acting on behalf of users.	-
security audit	An independent review and examination of system records and activities in order to test for adequacy of system controls, to ensure compliance with established policy and operational procedures, to detect breaches in security, and to recommend any indicated changes in control, policy and procedures.	sikringsrevisjon
security policy	The set of criteria for the provision of security services (see also identity-based and rule-based security policy). *Note* – A complete security policy will necessarily address many concerns which are outside of the scope of OSI.	sikringsreglar, regelsett for datasikring
security service	A service, provided by a layer of communicating open systems, which ensures adequate security of the systems or of data transfers.	sikringsteneste
selective field protection	The protection of specific fields within a message which is to be transmitted.	vern av utvalde felt
sensitivity	The characteristic of a resource which implies its value or importance, and may include its vulnerability.	sensitivitet
signature	See digital signature.	signatur
specialization	The technique of deriving a new managed object class from one or more existing managed object classes by inheritance and by the addition of new characteristics.	spesialisering
subclass	A class derived from another class by specialization	subklasse
superclass	A class used in deriving another class by specialization	superklasse
threat	A potential violation of security.	trugsmål
traffic analysis	The inference of information from observation of traffic flows (presence, absence, amount, direction and frequency).	trafikkanalyse
traffic flow confidentiality	A confidentiality service to protect against traffic analysis	konfidensialitet på trafikkflyt
traffic padding	The generation of spurious instances of communication, spurious data units and/or spurious data within data units.	trafikkfylling
trusted functionality	Functionality perceived to be correct with respect to some criteria, e.g. as established by a security policy.	-

Tabell A.1 Termear og definsjonar frå ITU-T-rekommendasjonar

A.2 Definisjonar frå IETF-spesifikasjonar

Term frå IETF-spesifikasjonar	**Definisjon frå IETF-spesifikasjonar**	**Term nytta i denne rapporten**
authentication scheme	The set of rules by which an SNMP message is identified as an authentic SNMP message for a particular SNMP community (33).	autentiseringsplan
authentication service	An implementation of a function that identifies authentic SNMP messages accoridin to one or more authentication schemes (33).	autentiseringsteneste
proxy agent	The SNMP agent associated with a proxy access policy	proxy-agent
SNMP access mode	An element of the set { read-only, read-write }	SNMP tilgangsmåte
SNMP access policy	A pairing of a SNMP community with a SNMP community profile	SNMP tilgangsreglar
SNMP application entity	The entities residing at management stations and network elements wich communicate with one another using the SNMP	SNMP applikasjonsentitet
SNMP authentic message	An SNMP message originated by an SNMP application entity that in fact belongs to the SNMP community naamed by the community component of said message (33).	autentisk SNMP-melding
SNMP community	A pairing of an SNMP agent with some arbitrary set of SNMP application entities (33).	SNMP fellesskap
SNMP community profile	A pairing of a SNMP access mode with a SNMP MIB view	SNMP fellesskapsprofil
SNMP MIB view	A subset of objects in the MIB that pertain to that element. (Note that the names of the object types represented in a SNMP MIB view need not belong to a single sub-tree of the object type name space	SNMP MIB-utsnitt
SNMP protocol entity	The peer processes which implement the SNMP and thus support the SNMP application entities (33).	SNMP protokollentitet
SNMP proxy access policy	For everey SNMP access policy: if the network element on which the SNMP agent for the specific SNMP community resides is not that to which the MIB view for the specific profile pertains.	SNMP proxy tilgangsreglar

Tabell A.2 Termar og definisjonar frå IETF-spesifikasjonar

B PROTOKOLLPROFILAR FOR TMN

B.1 Lågare lags protokollprofilar

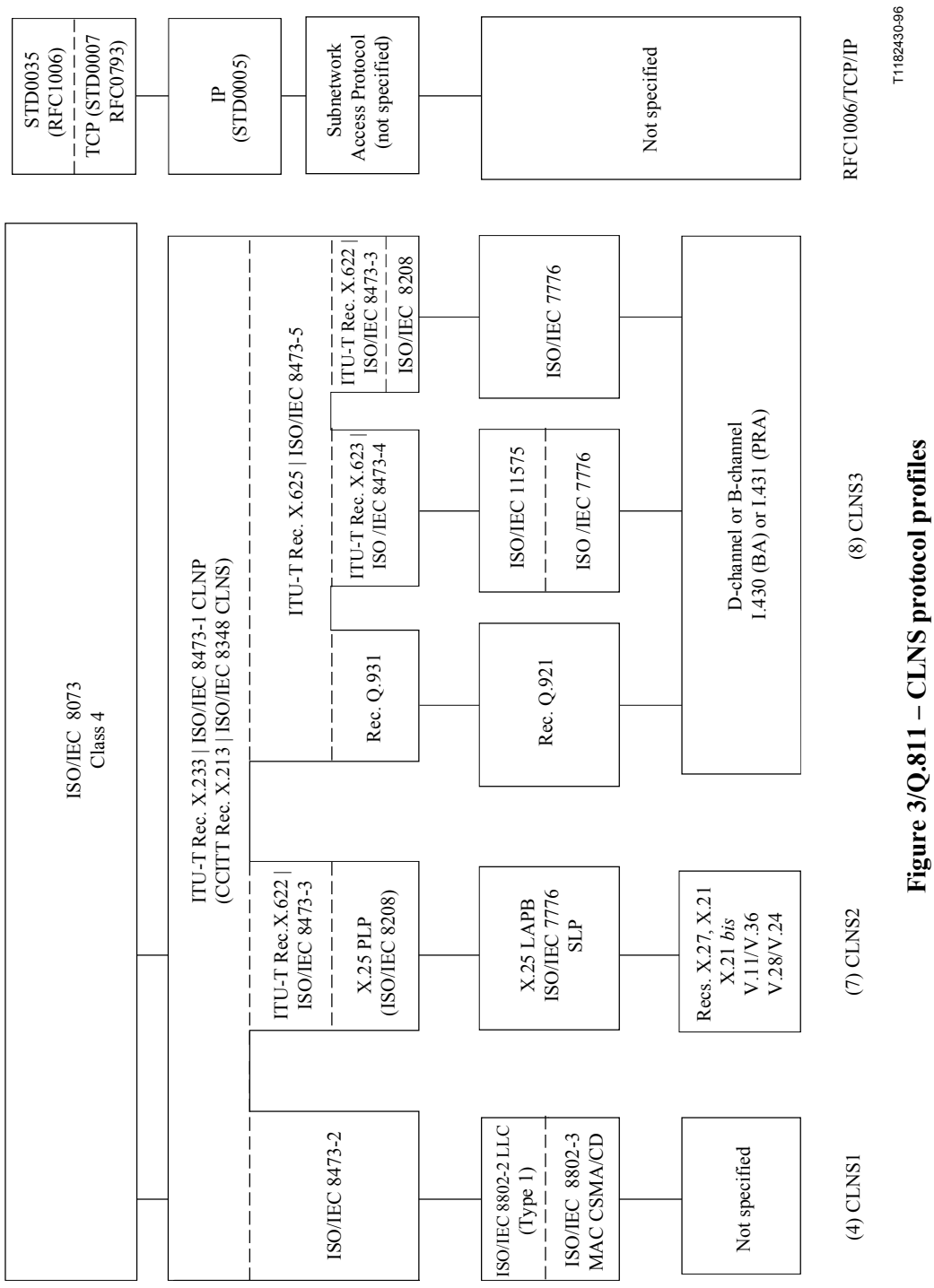
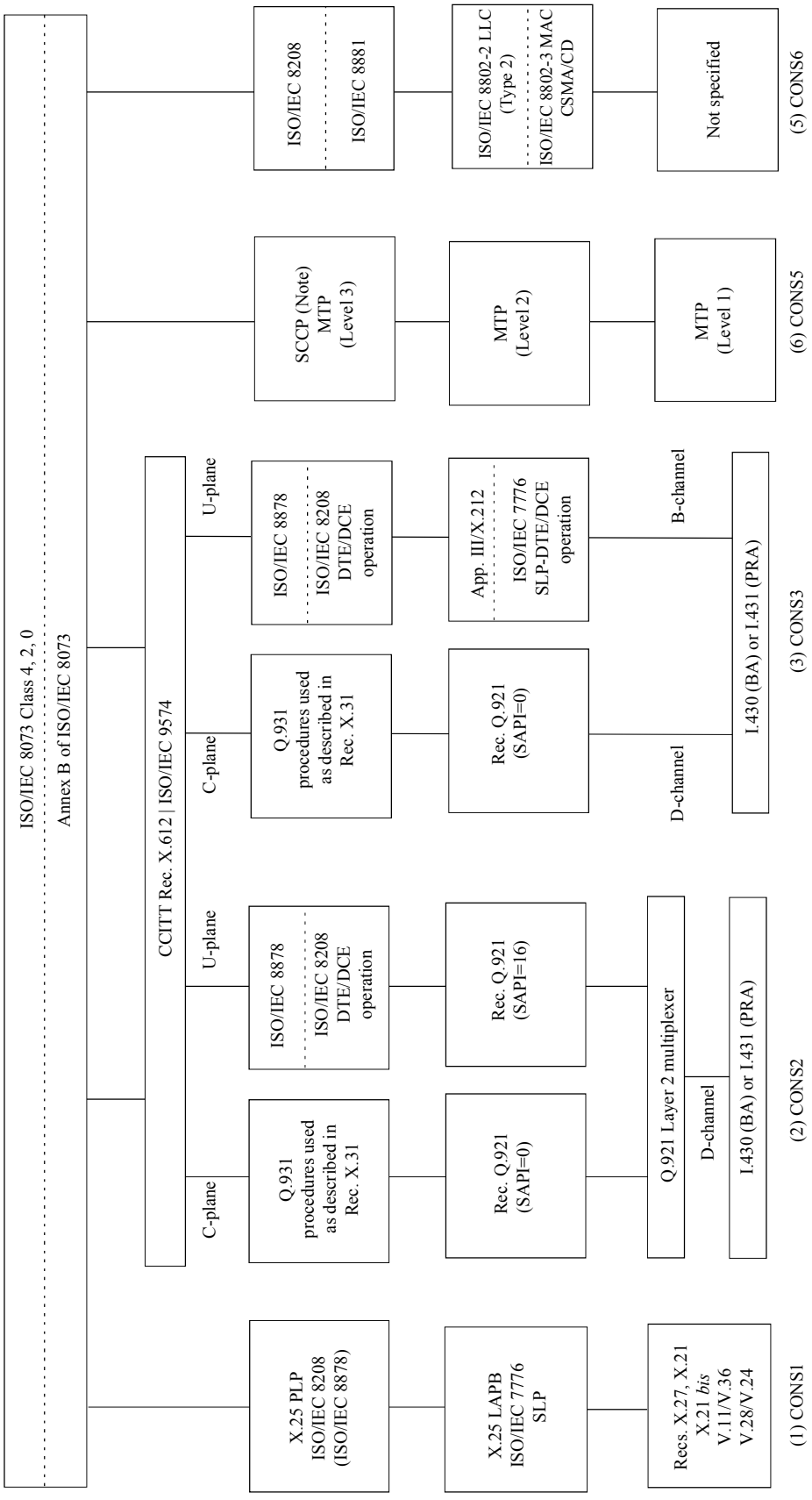


Figure 3/Q.811 – CLNS protocol profiles

Figur B.1 CLNS protokollprofilar(27)

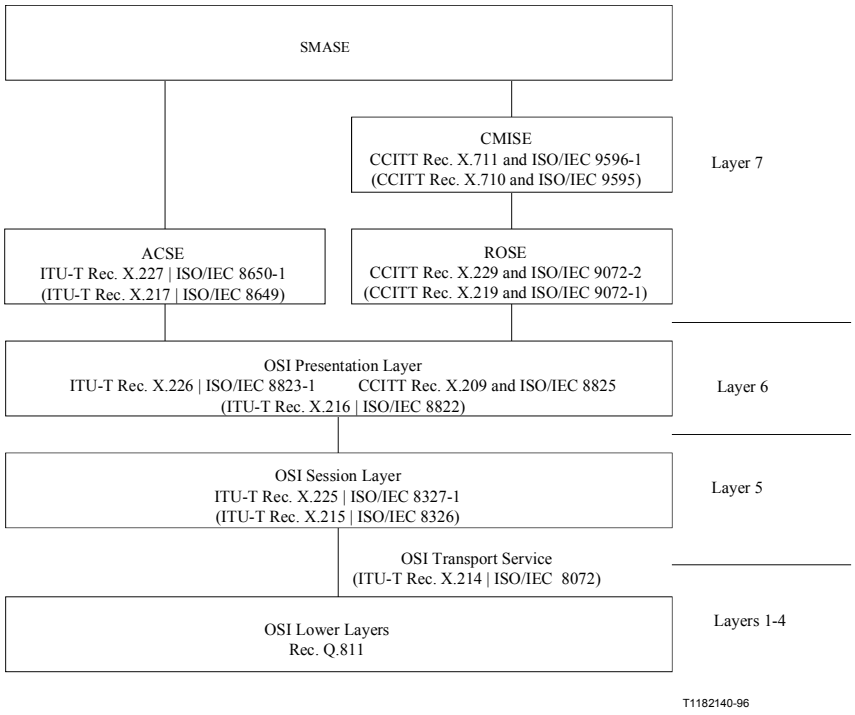


NOTE – Further study is needed for the function of SCCP at the boundary of Network layer and Transport layer.

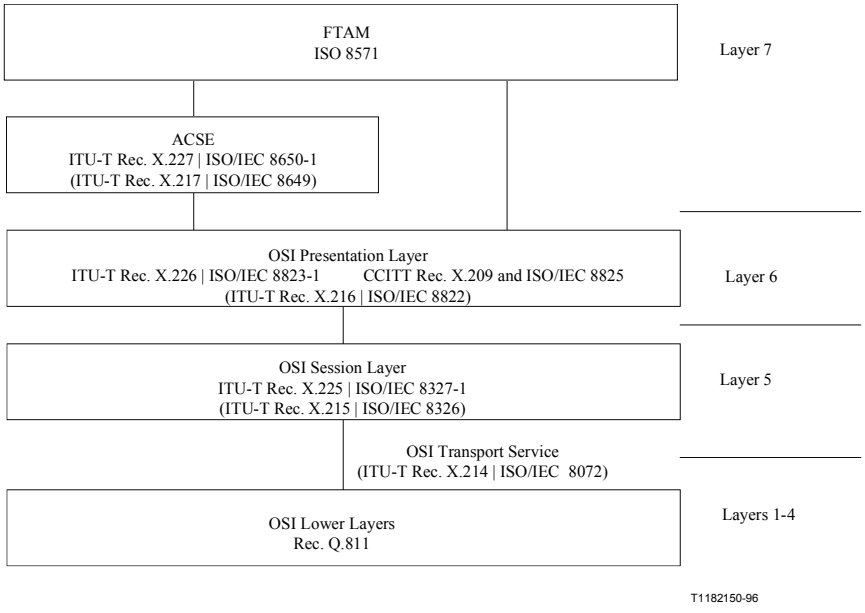
Figure 2/Q.811 – CONS protocol profiles

Figur B.2 CONS protokollprofilar (27)

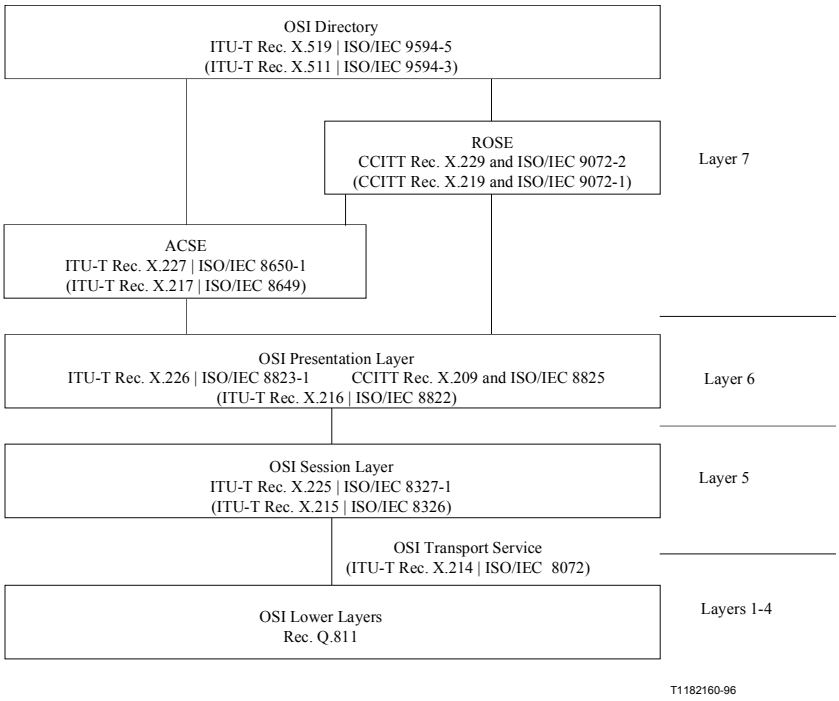
B.2 Høgare lags protokollprofilar



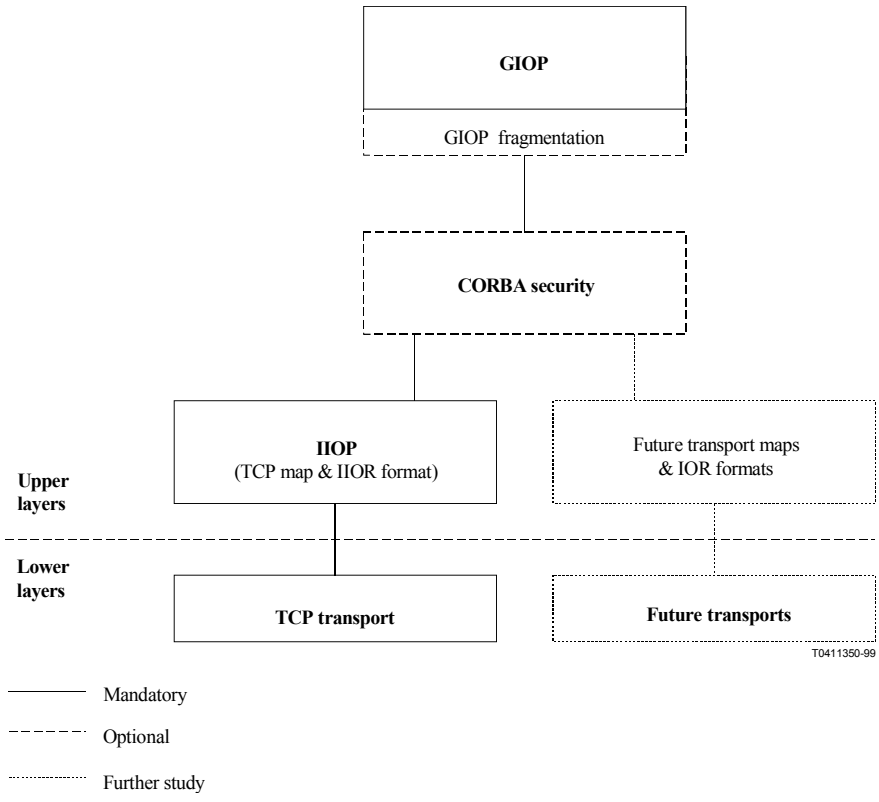
Figur B.3 Protokollprofil for interaktive tenester (28)



Figur B.4 Protokollprofil for filorienterte tenester (28)



Figur B.5 Protokollprofil for katalogtenester(28)



Figur B.6 CORBA-basert protokollprofil (28)

C FORKORTINGAR

ACSE	Assosiacion Control Service Element
AFU	Authetication Functional Unit
AgentX	Agent Extensibility
ANEP	Active Network Encapsulation Protocol
ASN.1	Abstract Syntax Notation One
ATM	Asynchronous Transfer Mode
CBC	Cipher Block Chaining
CIM	Common Information Model
CLNP	Connectionless-mode Network layer Protocol
CLNS	Connectionless-mode Network layer Service
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CMISE	Common Management Information Service Entity
COM	Component Object Model
CONS	Connection-mode Network layer Service
CORBA	Common Object Request Broker Architecture
CSMA/CD	Carrier Sense Multiple Access with Collosion Detection
DAP	Directory Access Protocol
DARPA	Defence Advanced Research Project Agency
DCF	Data Communication Functionality
DCN	Data Communication Network
DCOM	Distributed COM
DES	Data Encryption Standard
DME	Distributed Management Environment
DMTF	Distributed Management Tast Force
DN	Distinguished Name
DPE	Distributed Processing Environment
DSP	Directory System Protocol
EFD	Event Forwarding Discriminator
EGP	Exterior Gateway Protocol
ESIOP	Environment-Specific Inter-ORB Protocols
FR	Frame Relay
FTAM	File Transfer, Access and Management
FTP	File Transfer Protocol
GIOP	General Inter-ORB protocol
GNIM	Generic Network Information Model
HMAC	Hash Message Authentication Code
HMMP	HyperMedia Management Protocol
HTTP	HyperText Transfer Protocol
IAB	Internet Architecture Board
IDL	Interface Definition Language
IEC	International Electrotechnical Commissison
IETF	Internet Engineering Task Force
IIMC	ISO/CCITT and Internet Management Coexistence
IIOP	Internet Inter-ORB Protocol
IN	Intelligente Nett
IP	Internet Protocol
IPsec	IP security
IPX	Internetwork Packet Service
IRTF	Internet Research Task Force
ISDN	Integrated Services Digital Network
ISO	International Standardization Organisation
JIDM	Joint Inter-Domain Management
JMAPI	Java Management API

MAC	Medium Access Control
MAC	Message Authentication Code
MAF	Management Application Functionality
MbD	Management by Delegation
MIB	Mangement Information Base
MIT	Management Information Tree
MOF	Managed Object Format
MTP	Message Transfer Part
NE	Network Element
NEF	Network Element Function
NRIM	Network Resource Information Model
OIF	Optical Internetworking Forum
OMA	Object Management Architecture
OMG	Object Management Group
ORB	Object Request Broker
OS	Operations System
OSF	Open Software Foundation
OSF	Operation Systems Function
OSI	Open System Interconnection
OSIE	OSI Environment
OTN	Optical Transport Network
OTU	Optical Transport Unit
PDH	Plesiochronous Digital Hierarchy
PDU	Protocol Data Unit
PP	Presentation Protocol
PSTN	Public Switched Telephone Network
QA	Q-Adapter
QM	Q-Mediation
RAS	Reliability, Availability and Surviveability
RDN	Relative Distinguished Name
RFC	Requests for Comments
RMI	Remote Method Invocation
RMON	Remote Monitoring
ROSE	Remote Operation Service Element
RPC	Remote Procedure Call
SCCP	Signalling Connection Control Part
SDH	Synchronous Digital Hierarchy
SDU	Service DataUnit
SHA	Secure Hash Algorithm
SMI	Structure of Management Information
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SONET	Synchronous Optical Network
SP	Session Protocol
STM	Synchronous Transfer Mode
TCP	Transmission Control Protocol
TF	Transformation Function
TINA	Telecommunication Information Networking Architecture
TMF	Tele Management Forum
TMN	Telecommunication Mangement Network
TOM	Telecom Operation Map
UDP	User Datagram Protocol
USM	User-Based Security Model
VACM	View-Based Access Control Model
WBEM	Web-Based Enterprise Management
WS	Work Station
WSF	Work Station Function
XA	X-Adapter
XM	X-Mediation
XML	Extensible Markup Language

Litteratur

(1) Heinz-Gerd Hegering et al (1999): Integrated Management of Networked Systems, Morgan Kaufmann Publishers,USA.

(2) Morris Sloman (Ed.) (1994): Network and Distributed Systems Management, Addison-Wesley Publishing Company, England.

(3) William Stallings (1999): SNMP, SNMPv2,SNMPv3 and RMON 1 og 2, Addison Wesley, USA.

(4) David Zeltzerman (1999): A Practical Guide to SNMPv3 and Network Management, Prentice Hall PTR, USA.

(5) ITU-T (1994): Recommendation X.200 Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model.

(6) ITU-T (1995): Recommendation X.217 Information Technology - Open Systems Interconnection - Service definitions for the Association Control Service Element.

(7) CCITT (1988): Recommendation X.219 Remote Operations: Model, notation and service definition.

(8) ITU-T (1997): Recommendation X.519 Information Technology - Open Systems Interconnection - The Directory: Protocol specifications.

(9) CCITT (1992): Recommendation X.700 Management Framework for Open Systems Interconnection (OSI) for CCITT Applications.

(10) ITU-T (1997): Recommendation X.701 Information Technology - Open Systems Interconnection - Systems Management Overview.

(11) ITU-T (1997): Recommendation X.703 Information Technology - Open Distributed Management Architecture.

(12) ITU-T (1997): Recommendation X.710 Information Technology - Open Systems Interconnection - Common Management Information Service.

(13) ITU-T (1997): Recommendation X.711 Information Technology - Open Systems Interconnection - Common Management Information Protocol.

(14) CCITT (1992): Recommendation X.720 Information Technology - Open Systems Interconnection - Structure of Management Information: Management Information Model.

(15) CCITT (1992): Recommendation X.721 Information Technology - Open Systems Interconnection - Structure of Management Information: Definition of Management Information.

(16) CCITT (1992): Recommendation X.722 Information Technology - Open Systems Interconnection - Structure of Management Information: Guidelines for the Definition of Managed Objects.

(17) CCITT (1991): Recommendation X.800 Security Architecture for Open Systems Interconnection for CCITT Applications.

(18) ITU-T (2000): Recommendation M.3010 Telecommunication Management Network - Principles for a Telecommunications Management Network.

(19) ITU-T (2000): Recommendation M.3013 Considerations for Telecommunications Management Network.

(20) ITU-T (1998): Recommendation M.3016 Telecommunications Management Network - TMN Security Overview.

(21) ITU-T (1995): Recommendation M.3100 Telecommunications Management Network - Generic Network Information Model.

(22) ITU-T (1997): Recommendation M.3200 Telecommunications Management Network - TMN Management Services and Telecommunications managed areas: Overview.

(23) ITU-T (1998): Recommendation M.3300 Telecommunications Management Network - TMN F-interface requirements.

(24) ITU-T (1997): Recommendation M.3320 Telecommunications Management Network - Management requirements for the TMN X-interface.

(25) ITU-T (1997): Recommendation M.3400 Telecommunications Management Network - TMN Management Functions.

(26) ITU-T (1999): Recommendation G.872 Architecture of Optical Transport Networks.

(27) ITU-T (1997): Recommendation Q.811 Lower Layer protocol profiles for the Q3 and X interfaces.

(28) ITU-T (1997): Recommendation Q.812 Upper Layer protocol profiles for the Q3 and X interfaces.

(29) IETF (1990): RFC 1155 Structure and Identification of Management Information for TCP/IP-based Internets.

(30) IETF (1990): RFC 1156 Management Information Base for Network Management of TCP/IP-based Internets.

(31) IETF (1990): RFC 1157 A Simple Management Network Protocol (SNMP).

(32) IETF (1991): RFC 1212 Concise MIB Definitions.

(33) IETF (1991): RFC 1213 Management Information Base for Network Management of TCP/IP-based Internets.

(34) IETF (1993): RFC 1441 Introduction to version 2 of the Internet-standard Network Management Frameworkation for TCP/IP-based Internets.

(35) IETF (1996): RFC 1905 Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2).

(36) IETF (1996): RFC 1906 Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2).

(37) IETF (1996): RFC 1907 Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2).

(38) IETF (1997): RFC 2021 Remote Networking Monitoring Management Information Base Version 2 using SMIv2.

(39) IETF (1999): RFC 2274 User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3).

(40) IETF (1999): RFC 2275 View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP).

(41) IETF (2000): RFC 2276 Coexistence between Version 1, Version 2 and Version 3 of the Internet-standard Network Management Framework.

(42) IETF (1999): RFC 2571 An Architecture for describing SNMP Management Framework.

(43) IETF (1999): RFC 2573 SNMP Applications.

(44) IETF (1999): RFC 2578 Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2).

(45) IETF (1999): RFC 2579 Textual Conventions for SMIv2.

(46) IETF (1999): RFC 2580 Conformance Statements for SMIv2.

(47) IETF (2000): RFC 2741 Agent Extensibility (AgentX) Protocol Version 1.

(48) IETF (2000): RFC 2819 Remote Networking Monitoring Management Information Base.

(49) Ole-Erik Hedenstad et al (2001): Fremtidens telekommunikasjon - teknologiske og strukturelle trender, FFI/Rapport-2001/04498.

(50) Jan A. Audestad (1996): Introduction to Distributed Processing in Telecommunications Systems In: Lecture Notes, Course 45356, NTNU.

(51) William Stallings (1998): SNMP and SNMPv2: The infrastructure for Network Management, *IEEE Communications Magazine*, 3, 37-43.

(52) P.F. McKee et al (1999): Research directions in ditributed systems, *BT Technology Journal* **17**, 2, 137-144.

(53) Alan McGuire et al (2001): Applications of Control Plane Technology to Dynamic Configuration Management, *IEEE Communications Magazine* **39**, 9, 94-99.

(54) Brian J. Wilson et al (2000): Multiwavelength Optical Networking Management and Control, *IEEE Journal of Lightwave Technology* **18**, 12, 2038-2057.

(55) Albert Rafel et al (2001): A New Functional Model for Management of Optical Transport Network Nodes, *IEEE Journal of Lightwave Technology* **19**, 6, 810-820.

(56) Kurt S. Silverman et al (2000): Toward a Vision for Network and Service Management, Bell Labs Technical Journal 5, 4, 21-30.

(57) George Pavlou (2000): Using Distributed Object Technologies in Telecommunications Network Management, *IEEE Journal on Selected Areas in Communications* **18**, 5, 644-653.

(58) Nicholas R. Jennings et al (1998): Applications of Intelligent Agents In: *Agent Technology Foundations, Applications and Markets* (Eds Nicholas R. Jennings), Springer-Verlag.

(59) Hyacinth S. Nwana et al (1999): A Perspective on Software Agents Research, *The Knowledge Engineering Review* **14**, 2, 1-18.

(60) Konstantinos Psounis (1999): Active Networks: Applications, Security, Safety, and Architectures, *IEEE Communications Surveys* **2**, 1.

(61) Stamatis Karnouskos (2001): Security implications of implementing active network infrastructures using agent technology, *Computer Networks* **36**, 1, 87-99.

FORDELINGSLISTE

FFIE **Dato:** 5 september 2001

RAPPORTTYPE (KRYSS AV)	RAPPORT NR.	REFERANSE	RAPPORTENS DATO
X RAPP ☐ NOTAT ☐ RR	2001/04331	FFIE/794/110	5 september 2001
RAPPORTENS BESKYTTELSESGRAD		ANTALL EKS UTSTEDT	ANTALL SIDER
UGRADERT		55	131
RAPPORTENS TITTEL		FORFATTER(E)	
ARKITEKTURAR OG STANDARDAR FOR DRIFT OG STYRING AV KOMMUNIKASJONSNETT		WINJUM Eli	
FORDELING GODKJENT AV FORSKNINGSSJEF		FORDELING GODKJENT AV AVDELINGSSJEF:	
Torleiv Maseng		Johnny Bardal	

EKSTERN FORDELING

ANTALL	EKS NR	TIL
2		Forsvarets Overkommando
		Postboks 193
		Alnabru bedriftsterminal
		0614 Oslo
1		v/Arne Dimmen
1		v/John Olav Skogås
2		FTD
1		v/Tore Beck
1		v/Sten Mossin
1		v/Knut Rønning
1		v/Ole Peder Nordheim
2		FO/I
1		FO/I v/Per Trygve Gundersen
1		FO/I v/Tom Juliussen
1		HFK
1		SFK
1		LFK
		www.ffi.no

INTERN FORDELING

ANTALL	EKS NR	TIL
14		FFI-Bibl
1		Adm direktør/stabssjef
1		FFIE
1		FFISYS
1		FFIBM
1		FFIN
1		Torleiv Maseng, FFIE
1		Stian Løvold, FFIE
1		Vidar S Andersen, FFIE
1		Tor Gjertsen, FFIE
1		Geir Hallingstad, FFIE
1		Ronny Windvik, FFIE
1		Svein Haavik, FFIE
1		Karsten Bråthen, FFIE
1		Anton B Leere, FFIE
1		Ian B Bednar, FFIE
1		Kjell Olav Nystuen, FFIE
1		Ole Erik Hedenstad, FFIE
1		Frode Johan Lillevold, FFIE
1		Eli Winjum, FFIE
5		Arkiv, FFIE
		FFI-veven

FFI-K1 Retningslinjer for fordeling og forsendelse er gitt i Oraklet, Bind I, Bestemmelser om publikasjoner for Forsvarets forskningsinstitutt, pkt 2 og 5. Benytt ny side om nødvendig.