Institutt for informatikk

Automatisk feilhåndtering i kommunikasjons-systemer ved hjelp av Bayesnettverk og mønstergjenkjenn-ingsmetoder.

Ole-Christoffer Granmo

Hovedfagsoppgave

4. februar 1999

Forord.

Denne hovedoppgaven er resultatet av arbeidet jeg har gjort dels ved Institutt for Informatikk, Universitetet i Oslo og dels ved Telenor Bedrift, Divisjon IT-Service fra september 1997 til februar 1999. Følgende personer har vært innvolvert på ulike plan i dette arbeidet: Min kone Ellen har med sin hengivenhet vært en uvurderlig støtte både i medgang og motgang. Barna mine Mikkel og Sofie har vært flinke til å få meg opp om morgenen og gjort hver dag trivelig. Videre har veilederen min Olav Lysne gitt meg presis og inspirerende veiledning. Til slutt har de ansatte ved Telenor Bedrift, Divisjon IT-Service, avdeling for Nettverksteknologi og da spesielt Jan-Olav Rolfsnes vært en spennende gjeng å jobbe sammen med.

Takk til dere alle!

Ole-Christoffer Granmo, 4-2-1999.

Kapittel 1: Automatisk feilhåndtering i kommunikasjonssystemer - en oversikt.

Kapittel 2: En metode for automatisk, global og adaptiv feillokalisering og omfangsbestemmelse i kommunikasjonssystemer.

Kapittel 1: Automatisk feilhåndtering i kommunikasjonssystemer - en oversikt.

Ole-Christoffer Granmo

Institutt for Informatikk

Universitetet i Oslo.

10/2/99

1 Innledning.

Dagens kommunikasjonssystemer blir stadig mer komplekse etter hvert som ny teknologi tas i bruk og nye tjenester integreres. Distribuerte systemer som gjør bruk av den underliggende teknologien og de underliggende tjenestene blir samtidig mer sammensatte. Konsekvensen er at tradisjonell manuell feilhåndtering ikke lenger strekker til. Dette skaper et behov for nye effektive systemer for feilhåndtering i kommunikasjonssystemer.

Kunnskapsbaserte metoder som fuzzy logic, "case"-basert resonnering (CBR), nevrale nettverk og ulike former for statistiske metoder åpner for nye typer effektive og automatiske feilhåndteringssystemer. På åttitallet baserte slike feilhåndteringssystemer seg på regelbaserte ekspertsystemer. De regelbaserte ekspertsystemene har flere svakheter som kan oppsummeres i tungrodd modellering og oppdatering av kunnskap samt stivbent håndtering av informasjon [8, 12]. I de siste årene har disse svakhetene samt stadig mer komplekse og dynamiske kommunikasjonssystemer ført til stor aktivitet rundt såkalte hybride kunnskapsbaserte feilhåndteringssystemer. I de hybride kunnskapsbaserte feilhåndteringssystemene settes ulike former for kunnskapsteknologi sammen for å oppnå tilpasningsdyktig og automatisk feilhåndtering slik et moderne kommunikasjonssystem krever. Dette har ført med seg flere nye spennende metoder som [5, 9].

I denne hovedoppgaven foreslås et automatisk feilhåndteringssystem basert på et distribuert Bayesnettverk for global feillokalisering og omfangsbestemmelse. Videre interakterer Bayesnettverket med adaptive feildeteksjons- og feilkorrigeringsagenter basert på en mønstergjenkjenningsmetode. Før vi går i gang med å utlede egenskapene til dette systemet ser vi i *del 2* på hvordan feilhåndtering utføres tradisjonelt. Deretter i *del 3* gis en kort oversikt over grunnleggende kunnskapsbaserte metoder. Dette danner utgangspunktet for å vurdere de mer eller mindre utradisjonelle metodene som beskrives i *del 4*. Til slutt i *del 5* gis en kort oversikt over feilhåndteringssystemet som foreslås. Dermed er grunnlaget lagt for de påfølgende kapitlene.

2 Bakgrunn - tradisjonell feilhåndtering i kommunikasjonssystemer.

Grovt sett består et kommunikasjonssystem av *prosesseringsressurser* og *kommunikasjonsressurser*. Prosesseringsressursene behandler og utveksler data via *prosesseringstjenester* som for eksempel et databaseoppslag eller en multimediaavspilling. Når *tjenestebrukeren* eller prosesseringsressursene har ulik lokasjon sørger kommunikasjonsressursene for nødvendige *dataoverføringstjenester*.

Det stilles gjerne krav til kvaliteten på tjenestene. Kravene kan for eksempel være knyttet til kapasitet, tilgjengelighet, feilrate og utførelsestid. Dette spesifiseres i "kontrakter" som inngås mellom tjenestebruker og *tjenestetilbyder* slik at man oppnår en felles forståelse av hva slags tjenester som skal leveres.

Tjenestetilbyders oppgave blir å opprette og opprettholde spesifiserte tjenester. En tjeneste opprettes ved å allokere eller innføre nødvendige prosesserings- og kommunikasjonsressurser i kommunikasjonssystemet. Tjenesten opprettholdes ved å sørge for at ressursene fungerer over tid. Opprettelsen består grovt sett i å spesifisere, modellere, implementere og teste ut tjenesten. Opprettholdelsen består blant annet i å håndtere ressursfeilene som oppstår. Dette innebærer deteksjon av feil via overvåkning, lokalisering av detekterte feil, omfangsbestemmelse, og til slutt korrigering av lokaliserte feil. Det er *feilhåndteringsprosessen* som er temaet for hovedoppgaven og i denne delen gis en kort oversikt over tradisjonell feilhåndtering.

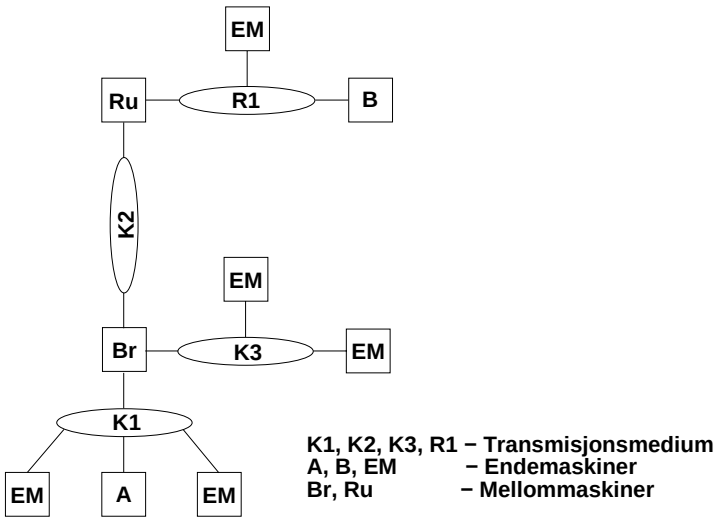
Før feilhåndteringsprosessen kan beskrives må leseren kjenne de grunnleggende funksjonene og komponentene i et kommunikasjonssystem. Under punkt *2.1* gis nødvendig bakgrunn. Deretter under *2.2* ser vi på hvordan ressursfeil i kommunikasjonssystemer tradisjonelt er håndtert. De tradisjonelle fremgangsmåtene har flere svakheter. Disse svakhetene ser vi på under *2.3*.

2.1 Kommunikasjonssystem.

Et kommunikasjonssystem består som nevnt av prosesseringsressurser og kommunikasjonsressurser. Prosesseringsressursene består videre av *applikasjoner* og *endemaskiner.* Kommunikasjonsressursene består av *kommunikasjonsprogramvare*, *mellommaskiner* og *transmisjonsmedier.*

Hardvare.

Det er endemaskiner, transmisjonsmedier og mellommaskiner som danner det fysiske grunnlaget for et kommunikasjonssystem. Endemaskinene stiller fysiske prosesseringsressurser til disposisjon for applikasjonene. Transmisjonsmediene sørger for fysisk overføring av bit mellom endemaskinene. Fordi ulike avstander og miljøer krever ulike typer transmisjonsmedier samt fordi kapasiteten til et transmisjonsmedium er begrenset, inneholder et kommunikasjonssystem gjerne flere transmisjonsmedier. Disse mediene knyttes sammen av mellommaskiner som dermed danner knutepunktene i kommunikasjonssystemet. En slik sammenknytning av maskiner kaller vi et *kommunikasjonsnettverk. Figur 1* gir et eksempel.



Figur 1: *Endemaskiner knyttet sammen via transmisjonsmedier og mellommaskiner.*

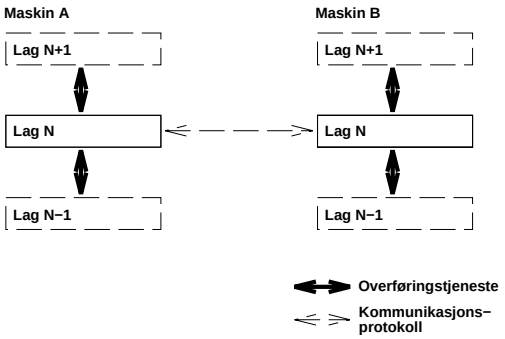
Å overføre bit fra en maskin til en annen i et kommunikasjonsnettverk er ikke trivielt. For det første kan transmisjonsmediene understøtte ulike grensesnitt mellom maskin og

medium, ulik koding av bit, ulik modulasjon av bit inn på en bærebølge og ulik bittakt. Videre må maskinene koordinere overføringen over et medium seg i mellom. Dette kan innebære deling av mediet, flytkontroll, feilkontroll, etc. Når i tillegg bit skal overføres mellom maskiner over flere transmisjonsmedier må overføringen koordineres med mellommaskinene. Dette krever global identifisering av maskiner, valg av vei gjennom nettverket samt samordning av forskjellige typer mellommaskiner og transmisjonsmedier. Denne kompleksiteten kan håndteres av programvare.

Programvare.

Applikasjoner og kommunikasjonsprogramvare danner grunnlaget for tjenestene i et kommunikasjonssystem. Applikasjonene behandler og utveksler data via prosesseringstjenester. Kommunikasjonsprogramvaren befinner seg i hver maskin og sørger for direkte kommunikasjon mellom applikasjonene ved å tilby overføringstjenester.

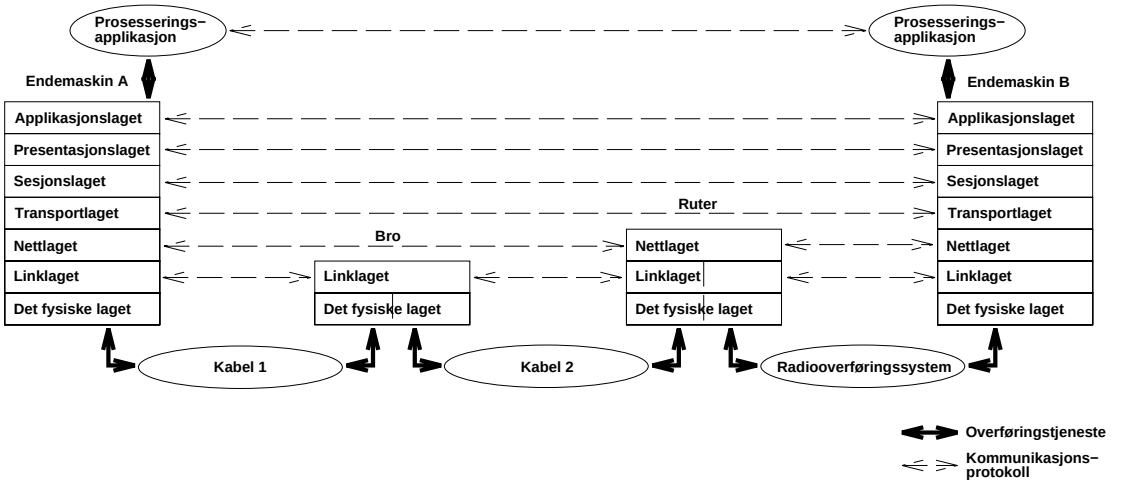
For å lettere kunne håndtere det beskrevne mangfoldet i et kommunikasjonsnettverk er kommunikasjonsprogramvare gjerne bygget opp lagvis i et hierarki slik *figur 2* illustrerer.



Figur 2: *Lag i kommunikasjonsprogramvaren.*

Lag *N* i maskin *A* samarbeider med lag *N* i maskin *B* om å tilby overføringstjenester mellom overliggende lag, *N+1*, i de respektive maskinene. Samarbeidet foregår etter gitte regler definert i en *kommunikasjonsprotokoll*, via overføringstjenestene til underliggende lag *N-1*. På denne måten abstraherer man stegvis bort forskjeller i hardvare og programvare. OSI-modellen er en referansemodell for slike hierarkier og gir oss en generell innfallsvinkel for feilhåndtering.

OSI-modellen har syv lag: det fysiske laget, linklaget, nettlaget, transportlaget, sesjonslaget, presentasjonslaget og applikasjonslaget. I vår sammenheng er ikke den nøyaktige funksjonen til hvert lag så viktig, men snarere at de er der og at de interakterer for å oppnå kommunikasjon mellom applikasjonene. Den interesserte leser kan få en grundig innføring i f.eks. [17]. Her gis det kun en forenklet skisse av hvordan lagene interakterer. Denne skissen illustreres med eksempelet i *figur 3*.



Figur 3: *Et eksempel på kommunikasjon mellom applikasjoner.*

I figuren representerer de stiplede pilene kommunikasjon mellom lag på samme nivå etter en gitt kommunikasjonsprotokoll. De ikke-stiplede pilene representerer bruk av overføringstjenester. Leseren bør merke seg sammenhengen mellom *figur 1*, *figur 2* og denne figuren.

Det laveste laget, som kalles *det fysiske laget*, tilbyr overføring av bit over et transmisjonsmedium mellom fysisk adskilte maskiner. Laget abstraherer bort medieavhengige aspekter som grensesnittet mellom maskin og medium (*interfacet*), koding av bit og modulasjon av bit inn på en bærebølge. Like transmisjonsmedier kan kobles sammen av *repeatere*. En repeater er en mellommaskin som "repeterer" bit ukritisk fra et transmisjonsmedium til et annet. Målet er å forsterke bærebølgen når en direkte sammenknytning gir for stor overføringsavstand for mediet.

Linklaget tilbyr pålitelig *forbindelsesorientert* eller *forbindelsesfri* overføring av bit mellom to maskiner. Dette oppnås ved å innføre en unik adresse for hver maskin, samt å ramme inn bitsekvenser med feilsjekkingsdata og kilde/destinasjonsadresse. Bitrammene overføres deretter via overføringstjenesten til det fysiske laget. Feilsjekkingsdataene brukes både ved forbindelsesorientert og forbindelsesfri overføring til å detektere *overføringsfeil*, dvs at bit er endret under overføringen. Ved forbindelsesfri overføring kaster linklaget rammer med feil. Dermed mottar laget over kun de feilfrie rammene. Ved forbindelsesorientert overføring unngås rammetap ved at linklagene i destinasjons- og kildemaskinen sørger for at rammer med feil overføres på nytt. Merk at konvensjonen med forbindelsesorientert og forbindelsesfri overføring går igjen på de neste lagene, men beskrives kun her.

Videre åpner adresseringen av maskiner for mer intelligente mellommaskiner, kalt *broer*, enn på det fysiske laget. Dette fordi adressene kan brukes til å kartlegge veier fra kildemaskin til destinasjonsmaskin. Når veiene er kartlagt velges en av veiene etter gitte kriterier. Broene overfører da rammer kun langs denne veien i stedet for globalt over alle transmisjonsmediene slik en repeater ville gjort det. Dermed reduseres belastningen på mediene. I enkelte tilfeller brukes også en bro til å oversette mellom to ulike typer fysiske lag.

Kommunikasjonsprotokollen, adresseringsformatet og rammeformatet som innføres i et linklag er ikke nødvendigvis uniformt innenfor kommunikasjonssystemet. I *figur 3* er for eksempel endemaskin *A* og "ruteren" knyttet til et lokalt og uniformt kommunikasjonsnettverk bestående av kabler og broer. Linklaget i endemaskin *A*, broen og ruteren

har da felles kommunikasjonsprotokoll, addresseringsformat og rammeformat. Endemaskin *B* er derimot koblet til et kommunikasjonsnettverk for overføring av bit over radiobølger. I dette kommunikasjonsnettverket har vi en helt annen kommunikasjonsprotokoll, adresseringsformat og rammeformat.

Nettlaget abstraherer bort slike forskjeller ved å tilby global overføring av data mellom endemaskiner via rutere (mellommaskiner). Ruterene fungerer som broer, men arbeider med globalt unike adresser innført på nettlagsnivået. Videre oversetter en ruter mellom to linklag i stedet for mellom to fysiske lag. I *figur 3* ser vi hvordan ruteren sørger for kommunikasjon mellom endemaskinene ved å oversette mellom linklagene til de to underliggende kommunikasjonsnettverkene.

Transportlaget er det første laget som er uavhengig av underliggende kommunikasjonsnettverk. I dette laget ønsker man å tilby overføringstjenester som kompenserer for varierende overføringskvalitet over kommunikasjonsnettverkene. På denne måten sørger laget for at overføringstjenesten fremstår som en uniform enhet for de applikasjonsorienterte lagene over.

Sesjonslaget tilbyr sesjoner mellom applikasjoner. En sesjon innebærer etablering av en transportforbindelse, overføring av data over denne og hensynsfull nedkobling etter bruk slik at alle data overføres korrekt. I tillegg tilbyr laget synkronisering og strukturering av overføringene.

Laget over sesjonslaget kalles *presentasjonslaget*. Presentasjonslaget sørger for kommunikasjon mellom applikasjoner som benytter ulik datasyntaks, dvs applikasjoner som behandler data med tilsvarende innhold, men som representerer innholdet på ulikt format. For det første forhandler presentasjonslaget i hver maskin seg frem til en felles overføringssyntaks slik at lagene kan snakke sammen. Hvis overføringssyntaksen er ulik datasyntaksen til applikasjonene, oversetter presentasjonslaget mellom disse. For det andre tilbyr laget overføringstjenester for sikring og kryptering av data.

En mellommaskin på dette nivået kan sørge for oversetting mellom to applikasjonsmiljøer hvor syntaksen er felles internt i hvert miljø men ulik mellom miljøene. Presentasjonsmellommaskiner kan også benyttes til å knytte lukkede miljøer sammen over et åpent miljø ved å sikre og kryptere overføringene.

Øverst i hierarkiet finner vi *applikasjonslaget*. Applikasjonslaget tilbyr basistjenester til applikasjonene. Basistjenestene kan for eksempel være filoverføring og filadministrasjon, dokumentoverføring og overføring av prosedyrekall.

Mellommaskiner på dette nivået sørger for sømløs integrering av applikasjoner uavhengig av lokasjonen i kommunikasjonssystemet.

To viktige egenskaper.

Vi har nå gitt en forenklet skisse av interaksjonen mellom lagene i OSI-modellen. Det er to basale egenskaper leseren bør merke seg.

For det første vil det ved en tjenesteutførelse aktiveres et sett av samarbeidende *komponenter*, dvs applikasjoner, kommunikasjonsprogramvarelag, maskiner og transmisjonsmedium. Vi kaller dette et *aktiveringssett*. I *figur 3* vil for eksempel en overføringstjeneste på transportnivå gi følgende aktiveringssett: transportlaget i *A*, nettlaget i *A*, linklaget i *A*, det fysiske laget i *A*, kabel 1, det fysiske laget i broen mot kabel 1, linklaget i broen, det fysiske laget i broen mot kabel 2, kabel 2, det fysiske laget i ruteren mot kabel 2, linklaget i ruteren mot kabelnettverket, nettlaget i ruteren, linklaget i ruteren mot radionettverket, det fysiske laget i ruteren mot radionettverket, radionettverket, det fysiske laget i *B*, linklaget i *B*, nettlaget i *B* og til slutt transportlaget i *B*.

Ved feilhåndtering kan det være nødvendig å analysere samarbeidet mellom komponentene i flere slike aktiveringssett parallelt. Dette er ingen triviell oppgave, men den andre basale egenskapen forenkler analysen noe.

Den andre basale egenskapen er at man ved å fjerne lag ovenfra og ned splitter kommunikasjonssystemet opp i *subkommunikasjonssystemer.* Disse subkommunikasjonssystemene kan analyseres hver for seg så lenge laget over analyseres helhetlig. Med helhetlig menes at komponentene og samarbeidene settes i sammenheng med hverandre. Anta for eksempel at interaksjonen mellom komponentene i en overføring fra applikasjonen i maskin *A* til applikasjonen i maskin *B* skal analyseres. Da kan man først analysere interaksjonen mellom lagene fra applikasjonsnivå til nettverksnivå helhetlig. Deretter kan interaksjonen fra linknivå til fysisk nivå analyseres separat i hvert av de to underliggende kommunikasjonsnettverkene fordi de danner to subkommunikasjonssystemer.

2.2 Tradisjonell feilhåndtering.

Feilhåndtering innebærer som nevnt deteksjon av feil via overvåkning, lokalisering av detekterte feil, omfangsbestemmelse og til slutt korrigering av lokaliserte feil. Følgende problemer må håndteres ved feilhåndtering:

- ❑ Hvordan kan man detektere alt fra innfløkte applikasjonsfeil til støyfeil på et transmisjonsmedium?
- ❑ Hvordan kan man lokalisere en gitt feil når komponentene interakterer i komplekse dynamiske sammensetninger slik at feil forplanter seg fra komponent til komponent?
- ❑ Hvordan kan man korrigere en gitt lokalisert feil når korrigeringen må skje på helt andre lokasjoner enn i feilpunktet?
- ❑ Hvordan håndterer man endringer i et kommunikasjonssystem når endringene medfører at svarene på foregående spørsmål må revurderes?

Under dette punktet ser vi på hvordan beskrevne problemer tradisjonelt er håndtert. Vi tar for oss funksjonen til hvert av feilhåndteringstrinnene samt hvordan hvert trinn tradisjonelt utføres.

Feildeteksjon.

Et kommunikasjonssystem overvåkes ved å måle *tjenesteattributter* og *komponentattributter.* En tjenesteattributt beskriver egenskaper ved en tjeneste, for eksempel tjenestens tilgjengelighet eller utførelsestid. Belastningen på et transmisjonsmedium eller antall detekterte rammefeil i nettlaget til en ruter er eksempler på komponentattributter; attributtene beskriver egenskaper ved en komponent.

Poenget med målingene er at de skal gi et dekkende bilde av ressurstilstanden i kommunikasjonssystemet. Med dekkende menes at alle de kritiske egenskapene i kommunikasjonssystemet måles så hyppig at ikke resultatene er foreldet ved bruk. Hvis bildet er dekkende vil ressursfeil kunne detekteres via endringer i måleresultatene. Feildeteksjon består dermed i å analysere måleresultat for så å avgjøre om de indikerer feil. Problemet er at i mange kommunikasjonssystemer er antall relevante målinger overveldende stort.

Tradisjonelt håndteres det store antallet målinger via selvstendige *sensorer* i komponentene. Sensorene utfører målinger og enkel feildeteksjon lokalt. Dermed blir en uhåndterlig oppgave for *en* maskin forvandlet til en ubetydelig oppgave for hver av maskinene i kommunikasjonssystemet. Problemet er at måleresultater ofte må ses i en helhet for å skille mellom tilfeldige målendringer og målendringer skyldt faktiske feil. Viktige måleresultater samles derfor i sentrale *innsamlingsstasjoner* for helhetlig analyse.

Måleresultater i innsamlingstasjoner danner på denne måten beslutningsgrunnlaget for feildeteksjon, enten ved automatisk sammenligning med predefinerte terskelverdier eller ved presentasjon i grafiske og tekstlige oversikter, lett tilgjengelig for analyse og sammenligning. Slike oversikter sammen med *terskelalarmene* danner utgangspunktet for feillokalisering. Med en terskelalarm menes alarmen som genereres når et måleresultat overskrider en terskelverdi.

Feillokalisering.

En detektert feil må lokaliseres før den kan korrigeres. Anta for eksempel at interaksjonen mellom nettlaget og linklaget i en kritisk ruter bryter sammen uten at sensorene i ruteren detekterer dette. Feilen vil blant annet forplante seg til applikasjonene som kommuniserer over det feilede leddet. Det betyr at man måler en rekke utilgjengelige tjenester, men ikke nødvendigvis direkte feil i ruteren. Dermed må informasjonsfragmentene samordnes for å lokalisere feilen. Dette innebærer i grove trekk å finne en eller flere feilkilder som sammen forklarer måleresultatene. Man må med andre ord kjenne sammenhengen mellom tilstanden til tjenester, aktiveringssett, komponentsamarbeid og komponenter knyttet til resultatet av målingene som utføres.

Tradisjonelt analyserer menneskelige *systemadministratorer* de grafiske og tekstlige oversiktene sammen med terskelalarmene. Med en systemadministrator mener vi en person med ansvaret for den daglige driften av et kommunikasjonssystem. Oversiktene forenkler feillokaliseringen ved at de hjelper systemadministratorene å følge årsakssammenhengene i kommunikasjonssystemet. Terskelalarmene retter systemadministratorens oppmerksomhet mot viktige måleresultater. Når oversiktene og terskelalarmene ikke gir tilstrekkelig informasjon må detekterte feil forfølges ved en samordnet analyse av tilstanden til tjenester, aktiveringssett, komponentsamarbeid, komponenter og nye manuelt utførte målinger. Denne jobben utføres gjerne av *systemeksperter* innenfor området. Med en systemekspert mener vi en spesialist innefor et gitt problemområde og dermed ofte en begrenset ressurs.

Omfangsbestemmelse.

Når en eller flere feil lokaliseres er det viktig å estimere omfanget av feilene. Dette for å kunne prioritere håndteringen av feil; en feil som kun medfører adaptiv omorganisering av ressurser er langt mindre alvorlig enn en feil som blokkerer en eller flere tjenester. Slike vurderinger krever inngående kjennskap til årsaksammenhengene mellom ressursfeil og tilstanden til tjenestene.

I et komplekst og stort kommunikasjonssystem er det gjerne kun systemeksperter innenfor feilområdene som kan utføre slike prioriteringer.

Feilkorrigering.

Etter at lokaliserte feil er rangert må det avgjøres hvordan feilene skal korrigeres. Korrigerende handlinger kan grovt sett klassifiseres som følger etter stigende kostnad: omkonfigurering og omstart av komponenter, oppgradering av komponenter og til slutt fysisk omkonfigurering eller utvidelser i kommunikasjonsnettverket. De fleste feil kan korrigeres ved omkonfigurering og omstart av komponentene.

Fordi det er systemekspertene som gjerne lokaliserer og bestemmer omfanget til feil, er det også ofte systemeksperten som utfører feilkorrigeringen. Å avgjøre den helhetlige effekten av handlinger mot en gitt komponent krever inngående kjennskap til årsakssammenhengene i kommunikasjonssystemet. Ved kompliserte feil er det derfor klart hensiktsmessige at eksperten utfører korrigeringen. Men ofte kan en feilet komponent sparkes i gang ved for eksempel en enkel restart. Da kunne også en systemadministrator ha korri-

gert feilene. Vi får altså en ressursmessig uheldig konsentrasjon av oppgaver hos systemekspertene.

Oppsummering.

Vi har nå beskrevet de ulike trinnene ved feilhåndtering og hvordan hvert trinn utføres tradisjonelt. Den beskrevne prosessen er iterativ og involverer gjerne en rekke mennesker. Vi ser på følgen av dette under neste punkt.

2.3 Svakheter ved tradisjonell feilhåndtering.

Fordi feilhåndteringsprosessen er iterativ og fordi den gjerne involverer en rekke menneskelige systemadministratorer og systemeksperter, får vi en feilhåndteringsflaskehals i store og komplekse kommunikasjonssystemer. En manuell analyse av tilstanden i et kommunikasjonssystem er i seg selv tidkrevende. Når den nødvendige kunnskapen i tillegg er fordelt mellom flere systemadministratorer og systemeksperter eskaleres flaskehalsen ytterligere. Det er derfor ønskelig å automatisere deler av systemadministratorenes og systemekspertenes oppgaver i den tradisjonelle feilhåndteringsprosessen.

I neste del ser vi på grunnleggende kunnskapsbaserte metoder som muliggjør slik automatisering.

3 Grunnleggende kunnskapsbaserte metoder.

Vi har sett at menneskelige systemadministratorer og systemeksperter skaper en flaskehals i feilhåndteringsprosessen når størrelsen og kompleksiteten i kommunikasjonssystemet blir for stor. Ved å automatisere hele eller deler av feilhåndteringen kan denne flaskehalsen løses opp.

Det er en rekke egenskaper ved feilhåndtering som taler for bruk av kunnskapsbaserte metoder. For det første krever feilhåndteringen innfløkte resonnementer rundt tilstanden til tjenester, aktiveringssett, komponentsamarbeid og komponenter knyttet til resultatet av målinger. Videre krever feilhåndteringen ofte at beslutninger tas på et uklart eller motstridende grunnlag. Dette er tilfellet når viktige målinger mangler ved tilgjengelighetsproblemer, eller når måleresultatene gir motstridende informasjon på grunn av tilfeldige målefeil eller manglende synkronisering. Totalt blir feilhåndtering svært vanskelig, så vanskelig at det som nevnt lenge har vært mennesker som har gjort størsteparten av jobben.

For å håndtere de beskrevne problemene har det vist seg gunstig å kombinere forskjellige kunnskapsbaserte metoder. Dette ser vi i de mer eller mindre utradisjonelle feilhåndteringssystemene som beskrives i *del 4*. Her følger en kort gjennomgang av noen grunnleggende metoder.

3.1 Logikkbaserte metoder.

I logikkbaserte metoder blir kunnskap representert i en kunnskapsbase som setninger uttrykt i et logisk språk. Uttrykkskraften i språket avgjøres av hvilket logisk system som benyttes. Utsagnslogikk gir relativt liten uttrykkskraft. Den sier bare noe om sannhetsverdien til setninger bygget opp av boolske konnektiver og symboler som representerer enkle påstander. Første ordens logikk er langt kraftigere ved at den deler verden opp i objekter og relasjoner mellom objektene. I tillegg benyttes kvantorer for å uttrykke kunnskap om setningers oppfyllbarhet eller allmenngyldighet. Andre logiske system har enda større uttrykkskraft ved at man for eksempel kan uttrykke resonnementer om tid implisitt i språket.

I logikkbaserte metoder resonnerer man ved å utlede nye setninger fra setningene i kunnskapsbasen. Nye setninger utledes ved hjelp av regler for logisk inferens. En viktig egenskap ved logisk inferens er at man kan utføre gyldige resonnementer bare ved å ta

hensyn til en delmengde av kunnskapsbasen. I tillegg forandres ikke gyldigheten til allerede kjente setninger når nye setninger tilføres kunnskapsbasen. Hvis dette ikke hadde vært tilfelle, ville kompleksiteten til eventuelle metoder for logisk inferens bli altfor stor.

Fordelen med logikkbaserte metoder er at man kan uttrykke og utlede presis og bevisbar kunnskap. Ulempen ved logikkbaserte metoder er at det ikke tas hensyn til usikkerhet. Setninger er enten sanne eller usanne. Ved å benytte sannsynlighetsverdier eller fuzzy logic for å uttrykke usikkerhet (se [15] for en oversikt over fuzzy logic), innfører man samtidig årsakssammenhenger som virker mellom setningene. Dermed holder ikke lenger de to egenskapene nevnt i forrige avsnitt. Det er mulig å uttrykke usikkerhet i begrensede tilfeller uten å ta hensyn til årsakssammenhengene, men man risikerer å få gå gale verdier. Systemer som benytter usikkerhet på denne måten (fuzzy logic) kalles ekstensjonelle systemer (se [18]).

En logikkbasert metode egner seg for automatisering av feilhåndtering i tilfeller hvor man har relativt sikre data og vil utlede presise resonnementer fra disse. Metoden egner seg ikke hvis inferenstrinnene er preget av usikkerhet og årsakssammenhengene er sammensatte.

I artikkel [4] brukes fuzzy logic til å bestemme korrigerende handlinger fra gitte måleresultater.

3.2 Metoder basert på sannsynlighetsteori.

I metoder basert på sannsynlighetsteori representeres kunnskap ved variable og variablenes sannsynlighetsfordeling. Ut fra den kombinerte sannsynlighetsfordelingen finner man sannsynligheten for alle mulige variabelinstansieringer. Bayes teorem (se [16] for en kort beskrivelse) kan brukes til å oppdatere sannsynlighetsfordelingene når ny kunnskap blir kjent. Inntil nylig har oppfatningen vært at sannsynlighetsteori ikke er håndterlig nok for generell bruk som kunnskapsbasert metode. Dette på grunn av antall variabelkonfigurasjoner vokser eksponensielt med antall variable. Nye kraftige metoder for å oppdatere og representere sannsynlighetsfordelinger i Bayesnettverk har forandret dette. I et Bayesnettverk modelleres den kombinerte sannsynlighetsfordelingen ved å definere styrken av årsakssammenhenger mellom variablene. Hver variabel gis en sannsynlighetsfordeling som beskriver følgen av ulike tilstander i direkte innfluerende variable. Se [13] for en nærmere beskrivelse av metode og begreper.

I et Bayesnettverk resonnerer man ved å beregne variablenes sannsynlighetsfordeling gitt alle kjente variabelinstansieringer. Den kombinerte sannsynlighetsfordelingen viser hvilken totaltilstand som er den mest sannsynlige.Ved å definere ønskeligheten av forskjellige variabelverdier kan man regne ut ønskeligheten av totaltilstanden. Man kan videre inkludere beslutningsvariable i modellen for å finne de beslutninger som med størst sannsynlighet vil føre til en ønsket tilstand.

Fordelen ved Bayesnettverk er at det tas hensyn til usikkerhet ved at årsakssammenhenger modelleres og håndteres eksplisitt og globalt, dvs intensjonelt (se [18]). Relativt til nevrale nettverk har vi i tillegg følgende fordeler. Bayesnettverk er semantisk forståelig ut fra feilområdet. Dermed er det mulig å bruke kjent kunnskap ved konstruksjon. Videre kan man trekke ut begrunnelser for resonnementene. Ulempen ved Bayesnettverk er imidlertid at utregningene er komplekse og visse strukturer er beregningsmessig uhåndterlige.

Ved feilhåndtering vil et Bayesnettverk være egnet når man må ta hensyn til usikkerhet og har oversikt over årsakssammenhengene mellom variablene. Hvis årsakssammenhengene er ukjente, kan man ta i bruk læringsmetoder for å generere Bayesnettverk automatisk. Dette krever at et godt eksempelmateriale er tilgjengelig. I en slik situasjon kan det i stedet være aktuelt å bruke nevrale nettverk som er beregningsmessig mer håndterlige.

I [5] brukes Dempfer-Shafer teorien (en ikke-Bayesiansk metode for håndtering av usikkerhet [19]) til å lokalisere feilede komponenter ved hjelp av tilgjengelighetsstatusen til komponentene fra et sentralt punkt i kommunikasjonssystemet. Bayesnettverk egner seg også godt i denne problemstillingen fordi man kan modellere følgefeilene som oppstår når tilgjengeligheten til komponenter måles fra et punkt.

3.3 Nevrale nettverk.

Et nevralt nettverk består av noder med en aktiveringsfunksjon og veide kanter mellom nodene. Nettverkets struktur bestemmer beregningskompleksiteten og representasjonsstyrken. I et feed-forward nettverk er kantene rettede og det er ingen sykler. I et lagdelt feed-forward nettverk er nodene arrangert i lag, hvor noder i hvert lag kun har kanter til nodene i neste lag. Noder uten foreldre er innvariable, og noder uten barn er utvariable. Nevrale nettverk av denne typen gir en funksjon som gitt innvariablene beregner utvariablene. Ved å endre vekten på kantene kan det nevrale nettverket tilpasse seg forskjellige funksjoner.

Et feed-forward nettverk representer kunnskap ved å finne generelle trekk i et eksempelmateriale. Ved feilberegninger under trening korrigeres nettverket ved hjelp av for eksempel Back-Propagation-algoritmen (se [14] for en beskrivelse av algoritmen). Algoritmen går baklengs gjennom nettverket og korrigerer vektene slik at nettverket tilpasser seg eksempeldata. Når nok eksempelmateriale er gjennomgått og eksempelmaterialet egner seg til analyse, vil det nevrale nettverket kunne beregne utvariable på egenhånd.

Styrken ved nevrale nettverk er at de kan trekke generell kunnskap om sammenhengen mellom variable fra et eksempelmateriale. Når et nettverk er trent er det svært tolerant overfor støy, og vil alltid finne best passende funksjonsverdi.

Svakheten ved nevrale nettverk er at de ikke er semantisk forståelige i forhold til problemområdet. Rent praktisk blir det dermed vanskelig for mennesker å forstå problemområdet og løsningsmetoden via det nevrale nettverket. Dermed kan man ikke finne hvilke begrunnelser et nettverk har for løsningen det kommer frem til. Man kan heller ikke generellt konstruere et nevralt nettverk manuelt fra eksisterende kunnskap.

Et nevralt nettverk er egnet til feilhåndtering i tilfeller hvor et godt eksempelmateriale er tilgjengelig og der de menneskelige ekspertene ikke har full oversikt over årsakssammenhengene innenfor problemområdet.

I artikkel [6] beskrives et problemområde hvor et nevralt nettverk kan brukes. Målet i artikkelen er å utvikle en metode for identifisering av permanente tilbakevendende nettverksfeil på et tidligst mulig tidspunkt. Metoden testes ut i AT&T's digitale kommunikasjonsnettverk. Her er det tilgjengelig et stort eksempelmateriale over nettverksfeil knyttet til måleresultater. Nettverksfeilene i eksempelmaterialet ble klassifisert i to grupper: forbigående feil, og permanente tilbakevendene feil. Spørsmålet var om man kunne finne forskjeller i måleresultatene mellom de to feiltypene ved feiltidspunktet. Her kan et nevralt nettverk benyttes til å finne eventuelle forskjeller ved å gå gjennom det grupperte eksempelmaterialet.

3.4 Modellbasert analyse.

I modellbasert analyse representeres kunnskap ved hjelp av en modell av analysemålet. Modellen kan bygges opp ved å se på enkeltkomponentenes funksjon og deretter sette dem sammen til en helhet.

Modellbasert analyse foregår ved at man prøver å forklare avvik mellom modellen og analysemålet. Avvikene forklares ved å endre tilstander i modellen til den oppfører seg tilsvarende analysemålet. Det er også mulig å forutsi hendelsesforløp, eller sette opp scenarier i modellen for videre analyse.

Fordelen ved en modellbasert analyse er at man får en konkret tolkning av analysemålet. Ved å holde modellen oppdatert med data fra analysemålet, kan man bruke modellen til å undersøke tilstander som ellers ikke er tilgjengelig. Ulempen er at resonneringskraften er relativt svak om man ikke kombinerer analysen med andre metoder.

I artikkel [5] ble en modell brukt til å trene opp et nevralt nettverk. Nettverket ble trent opp til å kjenne igjen flytproblemer på et tidlig stadium. Videre ble modellen benyttet til å svare på spørmål om dynamisk bestemte rutingveier.

For mer detaljert informasjon om kunnskapsbaserte metoder se [7]. I neste del tar vi for oss tre feilhåndteringssystemer som er basert på beskrevne metoder.

4 Eksisterende systemer for automatisk feilhåndtering.

Det finnes i dag en rekke regelbaserte ekspertsystemer som utfører mange av oppgavene som tradisjonelt har tilhørt systemadministratorer og systemeksperter. Målet med ekspertsystemene er å automatisere enkle oppgaver og støtte systemadministratorene og systemekspertene ved beslutningstagning i komplekse situasjoner.

De regelbaserte ekspertsystemene fungerer på høyde med menneskelige systemadministratorer og systemeksperter innenfor begrensede områder så lenge beslutningsgrunnlaget er klart og konsistent [12]. Som nevnt krever feilhåndtering ofte at beslutninger tas på et uklart eller motstridende grunnlag. Videre begrenses bredden og dybden på området til et ekspertsystem av krav om rask behandlingstid. Vi har også et problem ved endringer i kommunikasjonssystemet. Da kan kunnskap som er avhengig av sammensetningen til kommunikasjonssystemet utdateres. For å unngå dette må enten kunnskapen vedlikeholdes kontinuerlig eller være komponentbasert, dvs at sammensetningene i kommunikasjonssystemet ikke modelleres. Første tilfelle krever ofte manuell oppdatering, og siste tilfelle kan gi en lite helhetlig feilhåndtering. Totalt har vi dermed en rekke problemer som begrenser de regelbaserte ekspertsystemenes bruksområde.

Under følgende punkter se vi på tre mer eller mindre utradisjonelle feilhåndteringssystemer som baserer seg på mer moderne kunnskapsbaserte metoder. Likevel finner vi igjen flere av problemene.

4.1 Et fuzzy ekspertsystem for feilhåndtering.

I artikkel [4] beskrives et fuzzy ekspertsystem for feilhåndtering. Systemet foreslår en prioritering av korrigerende handlinger fra mottatte terskelalarmer og feilmeldinger. Kunnskapsbasen systemet baserer seg på er trukket ut av lagret feilhistorie, og representeres ved hjelp av fuzzy sett. Fuzzy settene danner maler som knytter feilkilder til terskelalarmer og feilmeldinger. Videre knyttes feilkildene til egnede korrigerende handlinger. Systemet har under laboratorieforsøk vist at det kan simulere en systemadministrators rolle ved feilhåndtering. Under forsøkene var korrekt korrigerende handling alltid blant systemets tre høyest prioriterte forslag. Artikkelen konkluderer dermed med at fuzzy ekspertsystem kan bli en verdifull støtte ved feilhåndtering i fremtiden.

4.2 Data Packet Network Expert Advisor.

I artikkel [9] beskrives systemet Data Packet Networks Expert Advisor som er et generisk verktøy for feilhåndtering. Systemet detekterer og lokaliserer feil fortløpende fra en strøm av terskelalarmer. Dette oppnås ved å sammenligne et sett av generelle feilmaler med terskelalarmene. Når en gitt feilmal passer til alarmene er feilen detektert og lokalisert. Feilmalene organiseres i et relasjonshierarki for å bedre oversikten til menneskelig systemadministratorer. Terskelalarmer som ikke faller inn under en mal samles automatisk slik at nye feiltyper kan identifiseres. Dermed danner de grunnlaget for manuell kon-

struksjon av nye maler. Systemet er svært raskt og håndterer 20 terskelalarmer per sekund. Resultatet er et effektivt feilhåndteringssystem hvor systemadministratoren får presentert abstraherte feilbeskrivelser fortløpende ved feiltilstander.

4.3 A Multiple Paradigme Diagnostic System for Wide Area Communication Networks.

I artikkel [5] beskrives et system for feillokalisering i WAN. Feillokaliseringssystemet bruker flere kunnskapsbaserte metoder i parallell; flere uavhengige enheter samarbeider for å lokalisere feil. I artikkelen beskrives fem enheter: En modell basert på Dempfer-Shafer teorien for tilgjengelighetsanalyse fra en gitt sentral innsamlingsstasjon. Denne enheten lokaliserer de fleste standardfeilene. Et regelbasert system utfører målinger ved spesielle feil samt verifiserer hypoteser foreslått av de andre enhetene. Videre benyttes en nettverkssimulator til å gjøre vei- og kandidatutvalg ved hypotesetesting. Et nevralt nettverk brukes til å oppdage feilmønstre på et tidlig stadium. De beskrevne enhetene samarbeider via en tavle.

Prototypen ble testet på et stort corporate WAN som spenner over flere kontinenter med mange mulige veier mellom flere tusen komponenter. Det er hovedsakelig rutere og broer som håndteres. Innebygd feiltolerans i ruteren og dynamisk ruting gjør feillokaliseringen vanskelig. Testresultat var likevel svært godt. I en testrekke på 2653 feil (i nettverkssimulatoren), identifiserte systemet alle feilene utenom en. Feilsituasjonene bestod av opp til 25 samtidige feil.

4.4 Diskusjon.

Alle systemene analyserer terskelalarmer og måleresultater i en abstrakt modell av kommunikasjonssystemet. Det som skiller systemene er inferensstyrken og detaljnivået i modellen samt hvor knyttet modellen er til sammensetningen av kommunikasjonssystemet.

I [4] håndteres uklare og inkonsistente terskelalarmer ved hjelp av fuzzy sett. Det beskrives derimot ingen mekanisme for håndtering av endringer i kommunikasjonssystemet. Settene som trekkes ut fra feilhistorien kan dermed bli utdatert.

Feilhåndteringssystemet i [9] er basert på “case”-basert resonnering [20]. Det er derfor naturlig å anta at systemet håndterer uklare og inkonsistente terskelalarmer. Det beskrives derimot ingen mekanisme som hindrer malene i å bli bundet til en gitt kommunikasjonssystemsammensetning. Vi får dermed et manuelt vedlikeholdsproblem både ved konstruksjonen av maler ved nye feiltyper og ved omfattende endringer i kommunikasjonssystemet.

Bortsett fra at systemet i [5] kun lokaliserer feil, ser det ut til å gi den mest helhetlige analysen av terskelalarmer og måleresultater. Ulempen med systemet synes å være kompleksiteten, kombinert med mangelen på distribusjon. Endringer i et kommunikasjonssystem kan også være et problem, spesielt for det nevrale nettverket. Men det er ikke beskrevet om og hvordan systemet håndterer endringer i artikkelen. Alt i alt virker systemet som en “testbed” for kunnskapsbasert feillokalisering hvor svært mange mer eller mindre overlappende metoder samkjøres med gode resultat.

Vi har som mål å oppnå en enhetlig og generisk håndtering av skisserte problemer som inkorporerer ønskelige egenskaper fra [4, 5, 9]. I neste del forslår vi et feilhåndteringssystem ved å kombinere et distribuert Bayesnettverk med en adaptiv mønstergjenkjenningsmetode. I resten av hovedoppgaven utleder og undersøker vi egenskaper ved dette feilhåndteringssystemet.

5 Forslag til et automatisk, distribuert og adaptivt feilhåndteringssystem.

Vi har nå identifisert en rekke problemer som må håndteres før man kan oppnå effektiv og automatisk feilhåndtering.

For det første har vi et ressursproblem. Ved feilhåndtering skal et overveldende antall tjenester og komponenter behandles detaljert og effektivt. Det viser seg at en sentralisert løsning ikke skalerer opp [1]; men en distribuert løsning er heller ikke uten problemer.

Vi får nemlig en lite intelligent feilhåndtering hvis tjenestene og komponentene kun behandles *lokalt*, dvs hver for seg. Dette fordi behandlingen av tjenester må ses i sammenheng med aktiveringssettene de bygger på og fordi samarbeidende komponenter må ses i sammenheng med hverandre; feilhåndteringen bør foregå på et *globalt* plan. Vi får dermed en konflikt mellom distribusjon og globalitet.

Problemene ved en global feilhåndtering stopper ikke med det. Globaliteten må nemlig representeres før man kan resonnere om den. Fordi et kommunikasjonssystem gjerne endres over tid vil en statisk representasjon raskt utdateres. Det er med andre ord ikke nok å finne en egnet representasjonsmodell og resonneringsmetode, representasjonsmodellen må også kunne tilpasse seg endringer i kommunikasjonssystemet.

I følgende kapitler forsøker vi å finne en løsning på disse problemene ved å foreslå et automatisk, distribuert og adaptivt feilhåndteringssystem. Feilhåndteringssystemet baseres på et distribuert Bayesnettverk for global feillokalisering og omfangsbestemmelse. Videre interakterer Bayesnettverket med feildeteksjons- og feilkorrigeringsagenter basert på en mønstergjenkjenningsmetode. Før vi går i gang med utledningen og undersøkelsene rundt dette feilhåndteringssystemet gir vi en kort oversikt.

5.1 Begrensninger.

Problemomfanget gjør at vi i første omgang begrenser oss til global feillokalisering og omfangsbestemmelse, mens feildeteksjonen og feilkorrigeringen foregår på et lokalt plan. Før vi går i gang med å beskrive sammensetningen til systemet tar vi for oss konsekvensen av disse begrensingene.

De fleste feil kan detekteres lokalt. Hvis en feil ikke kan detekteres lokalt, betyr dette at feilen kun detekteres ved å samordne måleresultater fra ulike tjenester og komponenter. Dermed vil en tjenestebruker heller ikke detektere feil ved utførelsen av en enkelt tjeneste. Lokalt ikke-detekterbare feil gir altså et minimalt utslag. Vi antar dermed foreløpig at lokal feildeteksjon er tilstrekkelig til å detektere de viktigste feilsituasjonene.

Mange feil må derimot korrigeres på et global plan, dvs at korrigeringen må planlegges og utføres ved å se hele kommunikasjonssystemet i ett. Men første trinn i en slik prosess er å bestemme feilsituasjonen globalt i kommunikasjonssystemet, dvs å lokalisere og bestemme omfanget av feil. Om ikke dette trinnet lar seg automatisere tilfredstillende, vil det heller ikke være mulig å oppnå global feilkorrigering. Vi nøyer oss derfor med å finne en metode for global feillokalisering og omfangsbestemmelse i første omgang.

På denne måten deles feilhåndteringsystemet i to. Et globalt feillokaliserings- og omfangsbestemmelsessystem og et lokalt feildeteksjons- og feilkorrigeringssystem. Vi tar her for oss førstnevnte, deretter ser vi på sistnevnte, og til slutt ser vi på interaksjonen mellom disse.

5.2 Global feillokalisering og omfangsbestemmelse.

For å oppnå global feillokalisering og omfangsbestemmelse velger vi å benytte et Bayesnettverk for modellering av årsakssammenhenger mellom tilstanden til tjenester, aktiveringssett, komponentsamarbeid og komponenter knyttet til resultatet av målinger. Et

Bayesnettverk håndterer modellering og resonnering med både deterministiske og stokastiske årsakssammenhenger. Dermed kan man med en egnet modell finne de mest sannsynlige tilstandene i kommunikasjonssystemet fra måleresultat selv ved manglende eller motstridene resultat. På denne måten bestemmer vi lokasjonen og omfanget til feil.

Det er tre hovedvanskeligheter som må håndteres. For det første trenger vi en egnet modell som inneholder relevante årsaksammenhenger. Når en slik modell er funnet kan vi konstruere statiske årsaksmodeller for et kommunikasjonssystem. Problemet er at en statisk årsaksmodell utdateres så snart kommunikasjonssystemet endres. Den andre vanskeligheten blir dermed å finne en metode for adaptiv tilpasning av årsaksmodellene. Til slutt har vi et generellt problem ved Bayesnettverk; store årsaksmodeller kan lett blir beregningsmessig uhåndterlige. Årsaksmodeller for kommunikasjonssystemer ser ut til å kunne bli både store og sammensatte. Vi må derfor undersøke om Bayesnettverkene vi konstruerer lar seg håndtere beregningsmessige, samt hvilke beregningsmetoder som er best egnet. I *kapittel 2* behandles disse problemene.

5.3 Lokal feildeteksjon og feilkorrigering.

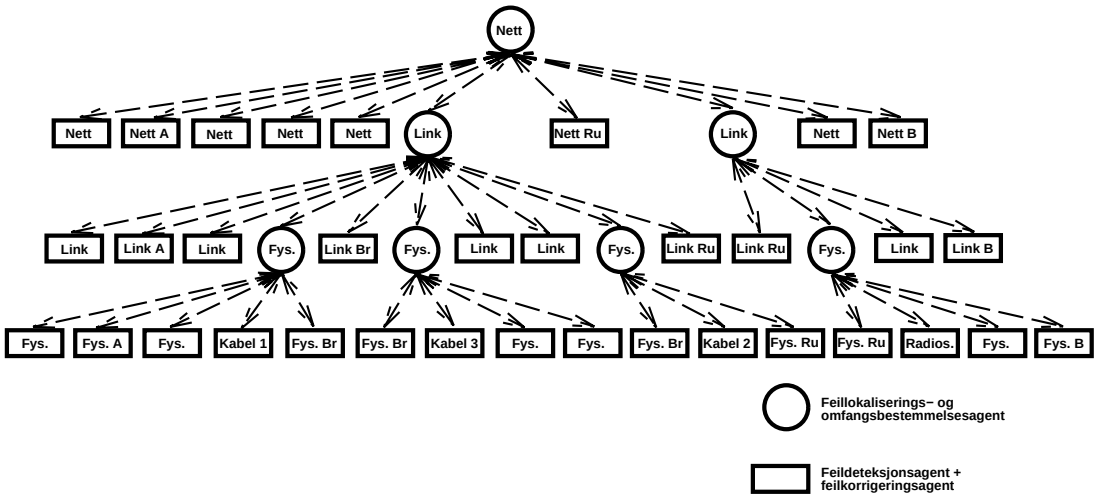
For å detektere feil lokalt i en komponent eller tjeneste må man kunne skille mellom feiltilstand og normal tilstand fra lokale måleresultater. For å korrigere en lokal feil bør man i tillegg avgjøre feilens type. Dette kan oppnås ved å lage en årsaksmodell som representerer komponentens interne struktur knyttet til måleresultat. Her har vi derimot mindre uniform kvalitativ kunnskap om årsaksammenhenger enn det vi har ved feillokalisering og omfangsbestemmelse. Vi foreslår derfor i stedet en metode for adaptiv direkte læring av koblingen mellom måleresultat og handling, dvs ved ren mønstergjenkjenning.

Følgende problemer skal håndteres. For det første må læringen være adaptiv med lokal tilpasning. Dette fordi måleresultater fra tilsvarende komponenter i ulikt miljø kan ha forskjellig tolkning. Videre kan tolkningen variere over tid etter hvert som et miljø forandrer seg. Dette krever både tilegning av nye tolkninger samt utvisking av gamle. Vi ønsker en kontrollert opplæring ved veiledning fra en menneskelig systemadministrator eler systemekspert. Målet er at man over tid skal få en feildeteksjons- og feilkorrigeringsdatabase som inneholder feildeteksjons- og feilkorrigeringserfaringer. For å oppnå effektiv feildeteksjon og feilkorrigering skal denne databasen distribueres ut til de aktuelle komponentene. Det må dermed utarbeides en protokoll for utveksling av erfaringer. I *kapittel 3* behandler vi disse problemene.

5.4 Et automatisk, distribuert og adaptivt feilhåndteringssystem.

For å oppnå skalerbarhet velger vi å distribuere feilhåndteringssystemet. Som vi så i *del 1* sørger den hierarkiske oppbygningen av kommunikasjonssprogramvaren for at vi rekursivt kan splitte kommunikasjonssystemet opp i selvstendige subkommunikasjonssystemer. Det betyr at subkommunikasjonssystemene kan behandles hver for seg så lenge sammenføyningen av subkommunikasjonssystemene behandles helhetlig. Vi benytter denne egenskapen til å foreslå et kontrollhierarki for adaptiv distribuert feilhåndtering.

Feilhåndteringen skal foretas av *agenter* på ulike abstraksjonsnivåer i kontrollhierarkiet slik *figur 4* illustrerer. Med en agent menes en enhet som iakttar omgivelsene ved hjelp av sensorer og utfører handlinger i omgivelsene ved hjelp av effektorer [7].



Figur 4: *Distribuert feilhåndteringssystem.*

Det underliggende kommunikasjonssystemet i figuren er eksempelkommunikasjonssystemet fra *del 1*.

På komponentnivå lar vi hver komponent behandles lokalt og detaljert av en *feildeteksjons- og feilkorrigeringsagent* bygget på resultatene fra *kapittel 3*. Disse agentene rapporterer detekterte feil til hierarkinivået over for global behandling. Videre utføres feilkorrigering ved kontroll fra overliggende hierarkinivå eller på agentens eget initiativ hvis kontrollen uteblir.

Fra fysisk nivå til applikasjonsnivå må tilstanden til tjenester, aktiveringssett, komponentsamarbeid og komponenter knyttet til resultatet av målinger analyseres for å danne et globalt bilde av tilstanden i kommunikasjonssystemet. En slik analyse kan utføres i Bayesnettverket som foreslås i *kapittel 2*. Resulterende tilstandsbilde danner grunnlaget for global kontroll av korrigeringen på komponentnivå.

Det er to problemer som må håndteres. For det første trenger vi en kommunikasjons- og samarbeidsprotokoll over kommunikasjonskanalene i kontrollhierarkiet. Et hovedmål er å oppnå adaptiv oppbygning av hierarkiet. For det andre fremstår et Bayesnettverk ved første øyekast som en atomisk enhet. Vi må finne en måte å distribuere Bayesnettverket i autonome enheter tilpasset kontrollhierarkiet. Målet er at ulike kontrollnivå skal kunne falle bort uten at den lokale feilhåndteringen på lavere nivå påvirkes. Dermed oppnås robust feilhåndtering. I *kapittel* 4 behandler vi disse problemene.

Til slutt i *kapittel 5* gis konklusjonen av arbeidet som er gjort.

6 Oppsummering.

I dette kapittelet ga vi først en oversikt over sammensetningen til kommunikasjonssystemer. Deretter beskrev vi generell feilhåndtering i et kommunikasjonssystem. Den beskrevne tradisjonelle fremgangsmåten for feilhåndtering har flere svakheter som kort kan oppsummeres i feilhåndteringsflaskehalsen de menneskelige systemadministratorene og systemekspertene danner.

Flere regelbaserte ekspertsystemer ble konstruert på åttitallet for å løse opp denne flaskehalsen. De regelbaserte ekspertsystemene har svakheter som tungrodd modellering og oppdatering av kunnskapsbasen samt begrenset håndtering av uklar og motstridene informasjon.

I de siste årene er det foreslått feilhåndteringssystemer som bøter på disse problemene ved å ta i bruk mer moderne kunnskapsbaserte metoder; deriblant nevrale nettverk, fuzzy logic og “case”-basert resonnering.

Global feilhåndtering, adaptivitet mhp på endringer i et kommunikasjonssystem og distribuerbarhet blir stadig viktigere ved håndtering av store, komplekse og dynamiske kommunikasjonssystemer. I denne hovedoppgaven foreslår vi et distribuert og adaptivt feilhåndteringssystem. Feilhåndteringssystemet er basert på et distribuert Bayesnettverk for global feillokalisering og omfangsbestemmelse i tett interaksjon med feildeteksjons- og feilkorrigeringsagenter basert på en mønstergjenkjenningsmetode. I de resterende kapitlene behandles dette feilhåndteringssystemet i detalj.

Da Bayesnettverket danner kjernen i feilhåndteringssystemet vil dette få den grundigste behandlingen. Vi undersøker egenskapene til Bayesnettverket ved hjelp av implementasjoner, eksperimentering og praktisk bruk. Feildeteksjonen, feilkorrigeringen og selve feilhåndteringssystemet behandles først og fremst ved å drøfte og utlede viktige egenskaper.

7 Referanser.

[1] Fergal Somers, HYBRID: Unifying Centralised and Distributed Network Management using Intelligent Agents, 1996 IEEE Network Operations and Management Symposium.

[2] Philip Steenekamp og Prof. J. Roos, Implementation of Service Management Policies: Applying Intelligent Agent Technologies, 1996 IEEE Network Operations and Management Symposium.

[3] Shyh-horng Jou og Shang-Juh Kao, An Autonomous Agent-Based Infrastructure for Inter-LAN Systems Management, 1997 IEEE Third International Symposium on Autonomous Decentralized Systems.

[4] Jiann-Liang Chen og Pei-Hwa Huang, A Fuzzy Expert System for Network Fault Management, 1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems.

[5] B. Pagurek, N. Dawes og R. Kaye, A Multiple Paradigme Diagnostic System for Wide Area Communication Networks, Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 1992.

[6] R. Sasisekharan, V. Seshadri og S. M. Weiss, Proactive Network Maintenance Using Machine Learning, 1993 IEEE Global Telecommunication Conference.

[7] Stuart Russel, Artificial intelligence: a modern approach.

[8] Morris Sloman, Network and Distributed Systems Management, s. 541-545.

[9] Sue Abu-Hakima, The DPN Expert Advisor: Canadian AI Success Story #3, Canadian Artificial Intelligence, 1994, nr 33, s. 14-19.

[10] Lundy Lewis, AI and Intelligent Networks in the 1990s and into the 21st Century, Worldwide Intelligent Systems, Approaches to Telecommunication and Network Management, 1995.

[11] Jill Huntington-Lee, Kornel Terplan og Jeff Gibson, HP OpenView, A Manager’s Guide, 1997, s. 127-128.

[12] O. Vermesan og A. I. Vermesan, The Use of Hybrid Intelligent Systems in Telecommunications, Worldwide Intelligent Systems, Approaches to Telecommunication and Network Management, 1995.

[13] Finn V. Jensen, An Introduction to Bayesian Networks, 1996, s. 7-29.

[14] Dr. Valluru B. Rao og Hayagriva V. Rao, C++ Neural Networks & Fuzzy Logic, s. 123-133, 1995.

[15] Toshiro Terano, Kiyoji Asai og Michio Sugeno, Applied Fuzzy Systems, 1994, s. 20-49.
[16] Gouri K. Bhattacharyya og Richard A. Johnson, Statistical Concepts and Methods, 1977, s. 87-95.
[17] Fred Halsall, Data communications, computer networks and open systems, Addison-Wesley, 1995.
[18] Judea Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Publishers, 1991, s. 3-14.
[19] Judea Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Publishers, 1991, s. 416-450.
[20] Ralph Bergmann, Case Based Reasoning on the Web, Miscellaneous, Introduction to Case-Based Reasoning, http://wwwagr.informatik.uni-kl.de/~lsa/CBR/index.html.

Kapittel 2: En metode for automatisk, global og adaptiv feillokalisering og omfangsbestemmelse i kommunikasjonssystemer.

Ole-Christoffer Granmo,

Institutt for Informatikk,

Universitetet i Oslo.

10/2/99

1 Innledning.

Mange av dagens kommunikasjonssystemer er svært komplekse. Tjenester komponentiseres, distribueres og flettes sammen i komplekse samvirkende systemer. Dette stiller krav både til prosesseringsressursene og kommunikasjonsressursene i et kommunikasjonssystem. Resultatet er at svært mange komponenter er involvert i opprettholdelsen av ønsket tjenestekvalitet i et komplekst samspill.

Denne kompleksiteten gjør det vanskelig å lokalisere feil. For det første kan feil forplante seg fra komponent til komponent i komplekse mønstere. Dermed tåkelegges feilkilden av en rekke *feilsymptomer*, dvs måleresultater som indikerer feil. For det andre detekteres feil ofte først flere ledd fra feilkilden. Da må feilkilden forfølges gjennom kommunikasjonssystemet.

Omfanget til en feil kan bestemmes ved å dedusere hvilken virkning feilen har på kommunikasjonssystemet. Dette er vanskelig når komponentene virker i komplekse samspill.

Disse problemene gjør tradisjonell manuell feilhåndtering både tidkrevende og kompetansekrevende. Vi har derfor utviklet en metode for automatisk feillokalisering og omfangsbestemmelse i kommunikasjonssystemer.

Metoden går ut på å automatisk generere et Bayesnettverk som modeller årsaksammenhengene mellom tilstanden til tjenester, aktiveringssett, komponentsamarbeid og komponenter (begrepene ble definert i *kapittel 1*). Dette Bayesnettverket brukes så til å lokalisere og bestemme omfanget av feil ved å beregne de mest sannsynlige tilstandene i kommunikasjonssystemet fra kjente tilstander.

Hypotesene i dette kapittelet er at

H1: Metoden gir global feillokalisering og omfangsbestemmelse.

H2: Metoden er adaptiv med hensyn på sammensetningen til et kommunikasjonssystem.

H3: Metoden lokaliserer og bestemmer omfanget til feil nøyaktig og effektivt.

Vi argumenterer her for at **H1** og **H2** er viktig for at **H3** skal holde. Deretter drøftes to problemer som vanskeliggjør oppnåelsen av **H3**. I resten av kapittelet utledes selve metoden.

Global feillokalisering og omfangsbestemmelse.

Med global feillokalisering og omfangsbestemmelse menes her at årsaksammenhengene mellom tilstanden til tjenester, aktiveringssett, komponentsamarbeid og komponenter mo-

delleres og håndteres globalt i kommunikasjonssystemet. Globalitet er viktig fordi en enkel tilstandsforandring kan forårsake tilstandsforandringer i hele kommunikasjonssystemet. Videre virker gjerne komponentene i et kommunikasjonssystem i komplekse samspill. Da holder det ikke å analysere virkningen av tilstanden til en tjeneste, et aktiveringssett, et komponentsamarbeid eller en komponent separat; interaksjonen mellom tilstandene til alle tjenester, aktiveringssett, komponentsamarbeid og komponenter må analyseres som en del av en samtidig helhet.

Alternativet er å modellere og håndtere et utvalgt sett av lokale årsakssammenhenger. Denne lokaliteten effektiviserer håndteringen, men i komplekse situasjoner får vi unøyaktig og i verste fall inkonsistent håndtering. Anta for eksempel at en feilsituasjon kan forklares av tre alternative feilkilder. Anta videre at årsaksammenhengene mellom feilkildene og feilsymptomene modellerer og håndterer separat for hver feilkilde. Da vil man kun identifisere de tre feilkildene men ikke at de er alternative forklaringer til de samme feilsymptomene.

Vi konkluderer derfor med at global modellering og håndtering av årsakssammenhenger er viktig for å oppnå mest mulig nøyaktig feillokalisering og omfangsbestemmelse.

Adaptivitet med hensyn på den strukturelle sammensetningen til et kommunikasjonssystem.

Kommunikasjonssystemer endrer seg gjerne over tid. Enten ved at den strukturelle sammensetningen (topologien) forandres eller ved at ny teknologi innføres. Denne typen endringer krever gjerne manuelle oppdateringer i tradisjonelle regelbaserte ekspertsystemer for nettverksadministrasjon generelt [1] og i ekspertsystemer for feillokalisering og omfangsbestemmelse spesielt. Vi får dermed en oppdateringsflaskehals hvis alvorlighet øker med størrelsen og endringsraten i kommunikasjonssystemet.

En feillokaliserings- og omfangsbestemmelsesmetode kan være adaptiv med hensyn på et kommunikasjonssystems strukturelle sammensetning. Da vil kun ny teknologi kreve manuell oppdatering. Dette er en forbedring fra en statisk fremgangsmåte fordi ny teknologi innføres langt sjeldnere enn den strukturelle sammensetningen i et kommunikasjonssystem endres.

I tillegg kan en feillokaliserings- og omfangsbestemmelsesmetode være adaptiv med hensyn på komponentenes virkemåte. Det betyr at ny teknologi håndteres automatisk.

Full adaptivitet er ønskelig, men ulempen er at adaptiviteten kan gi et uoversiktlig og muligens ustabilt system. Ingen adaptivitet gir lite effektiv eller lite nøyaktig feillokalisering og omfangsbestemmelse. Vi velger derfor middelveien; adaptivitet med hensyn på den strukturelle sammensetningen til et kommunikasjonssystem.

To hovedproblemer som vanskeliggjør feillokalisering og omfangsbestemmelse.

Det er to praktiske problemer som må håndteres i tillegg til globaliteten og endringsraten i et kommunikasjonssystem.

For det første kan viktige måleresultater være utilgjengelige. Enten på grunn av feil i i overføringstjenestene eller av sikkerhetsmessige grunner. Resultatet er at feillokaliseringen må skje på et uklart beslutningsgrunnlag.

For det andre kan måleresultater være motstridende. Enten på grunn av manglende synkronisering av målinger eller på grunn av *ikke-totale feil*. Med en ikke-total feil menes at feilen ikke gjør seg gjeldende ved hver eksponering, for eksempel ved en måling. Dermed må feil lokaliseres på et motstridende beslutningsgrunnlag.

Problemene begrenser bruksområdet til tradisjonelle regelbaserte ekspertsystemer. Vi skal undersøke hvordan et Bayesnettverk håndterer disse problemene.

Oversikt.

Først, i *del 2*, gis en kort beskrivelse av Telenors *TCP/IP-SNA* gatewaysystem. Dette kommunikasjonssystemet gir oss et realistisk testmiljø for metoden. Videre brukes eksempler fra dette kommunikasjonssystemet til å illustrere utledningene i kapittelet.

Vi trenger et generelt fundament å utlede metoden fra. I *del 3* foreslås et slikt fundament; et språk for spesifikasjon av sammensetningen til tjenester, aktiveringssett og komponentsamarbeid i et kommunikasjonssystem. Dette fundamentet er adaptivt fordi man relativt enkelt kan spesifisere slike sammensetninger automatisk.

Deretter, i *del 4*, foreslås et Bayesnettverk som modellerer årsaksammenhengene mellom tilstanden til tjenester, aktiveringssett, komponentsamarbeid og komponenter i et kommunikasjonssystem. Bayesnettverket bygger direkte på spesifikasjonsspråket fra *del 3* ved at det genereres automatisk fra strenger i språket. Vi argumenterer for at denne fremgangsmåten gir adaptiv feillokalisering og omfangsbestemmelse med hensyn på den strukturelle sammensetningen til et kommunikasjonssystem.

I *del 5* undersøkes det om genererte Bayesnettverk gir effektiv og nøyaktig feillokalisering og omgangsbestemmelse, samt hvilke beregningsmetoder som er best egnet. Dette gjør vi ved å undersøke nøyaktigheten og effektiviteten til beregningsmetodene i tilfeldig genererte feilsituasjoner.

Videre, i *del 6*, undersøkes feillokaliserings- og omfangsbestemmelsesmetodens praktiske verdi. Her beskriver vi en implementasjon av metoden i Telenors TCP/IP-SNA gatewaysystem. Vi diskuterer også resultatet av en fire måneder lang testperiode.

Til slutt, i *del 7*, gis en oppsummering av arbeidet samt en kort oversikt over videre arbeid.

2 Eksempelkommunikasjonssystemet.

Feillokaliserings- og omfangsbestemmelsesmetoden testes i Telenors TCP/IP-SNA gatewaysystem. Dette kommunikasjonssystemet sørger for sømløs overføring av data mellom klientapplikasjoner i et landsdekkende TCP/IP-kommunikasjonsnettverk og databaser i et separat *SNA-kommunikasjonsnettverk*. Med et SNA-kommunikasjonsnettverk menes her en spesiell nettverksløsning fra IBM. Kunnskap om denne nettverksløsningen er ikke nødvendig her.

Eksempelkommunikasjonssystemet understøtter en rekke applikasjoner. For det første understøttes et analysesystem som gir informasjon om det ordinære telefoninettets tilstand. Dette analysesystemet benyttes av Telenor Nett og har 200-300 brukere.

Videre understøttes en tjeneste for håndtering av alle henvendelser om feil på telefonitjenester som blir meldt på tlf. nr. 140 (feilmeldingen).

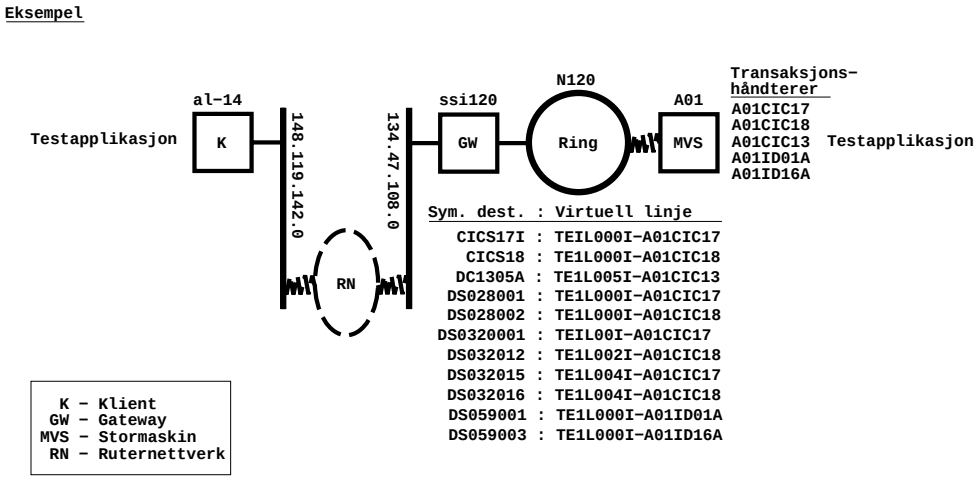
En av de mer kritiske systemene som understøttes er Sofa-systemet. Dette systemet aktiveres ved anrop på nødnummerene 110, 112 og 113. Sofa-systemet avgjør på grunnlag av telefonnummeret til den som ringer, navn, adresse og eventuelle kartkoordinater (for lokasjonen til benyttet telefon). I dette systemet er det forståelig nok nødvendig med oppetid nær 100%.

Det siste systemet vi beskriver benyttes av Telenor Privat for behandling av alle typer klager og henvendelser fra kunder. I tillegg registreres undersøkelser som skal utføres i tilknytning til en klage. Dette systemet benyttes av 500-750 brukere.

Krav om oppetid i beskrevne systemer gjør det viktig at lokasjonen og omfanget til feil kan bestemmes nøyaktig og effektivt.

2.1 Kommunikasjonssystemet.

Kommunikasjonssystemet består totalt av *89* klienter, *50* TCP/IP-segmenter, *10* gatewayer, *108* symbolske forbindelser, *91* virtuelle linjer, *4* ringer, *4* MVSer og *16* transaksjonshåndterere. *Figur 1* illustrerer en delløsning fra kommunikasjonssystemet. De delene av kommunikasjonsnettverkene som ikke er direkte relevante for problemstillingen er abstrahert bort.



Figur 1: *En delløsning innenfor Telenors TCP/IP-SNA gatewaysystem.*

Vi tar først for oss hardvaren. Deretter ser vi på en abstrahert skisse av programvaren.

Hver klientmaskin er knyttet til en gitt gateway (mellommaskin) via TCP/IP-kommunikasjonsnettverket. Videre er hver gateway knyttet til en ring i SNA-kommunikasjonsnettverket. SNA-nettverket knytter til slutt stormaskinene (MVSene) til ringene.

En klientapplikasjon kommuniserer med databasene i stormaskinene over symbolske forbindelser definert i gatewayene. I figuren vises noen symbolske forbindelser (destinasjoner) definert i gateway *ssi120*. Vi kan si at den symbolske overføringstjenesten tilbys av et kommunikasjonsprogramvarelag i klientmaskinene, gatewayene og MVSene. Laget mapper overføringstjenesten ned på TCP/IP-kommunikasjonsnettverket og SNA-kommunikasjonsnettverket med gatewayene som mellomledd.

På TCP/IP-siden tilbys en enkel forbindelsesorientert overføringstjeneste mellom klientmaskiner og gatewayer. På SNA-siden tilbys mer komplekse overføringstjenester over virtuelle linjer mellom gatewayer og MVSer. De virtuelle linjene er statisk definert både i kildemaskinene (gatewayene) og destinasjonsmaskinene (MVSene). Dette illustreres i figuren ved at de virtuelle linjene er knyttet til et gitt kildepunkt i gatewayene og et gitt destinasjonspunkt (en transaksjonshåndterer) i MVSene.

Vi ser nå på hvordan overføringstjenesten mellom klientapplikasjonene og databasene overvåkes.

2.2 Overvåkningen.

For å kunne overvåke forbindelsene mellom klientapplikasjonene og databasene er det installert en testapplikasjon på hver klientmaskin. Testapplikasjonene tester ut gitte symbolske forbindelser. En feilet test indikerer feil i en av komponentene i benyttede aktiveringssett. Ved å kjøre test-applikasjonen fra flere klienter mot forskjellige MVSer via ulike gatewayer, kan man danne seg et bilde av den globale tilstanden i kommunikasjonssystemet. Dette krever inngående kjennskap til årsakssammenhengene mellom til-

standen til tjenestene, aktiveringssettene, komponentsamarbeidene og komponentene i kommunikasjonssystemet.

2.3 Konklusjon.

Dette kommunikasjonssystemet gir oss et realistisk og komplekst miljø hvor vi kan undersøke egenskapene til feillokaliserings- og omfangsbestemmelsesmetoden. Videre danner *figur 1* utgangspunktet for eksemplene i kapittelet.

Vi tar nå for oss hvordan sammensetningen til et kommunikasjonssystem kan spesifiseres slik at vi får et solid fundament for feillokaliserings- og omfangsbestemmelsesmetoden.

3 Et språk for spesifikasjon av sammensetningen til tjenester i et kommunikasjonssystem.

For å kunne lokalisere og bestemme omfanget til feil må sammensetningen til tjenester, aktiveringssett og komponentsamarbeid være kjent. Slike sammensetninger kan ofte hentes direkte fra kommunikasjonssystemet eller fra konfigurasjons- og topologidatabaser (se f.eks. [2] for detaljer). Disse informasjonskildene strukturerer og presenterer gjerne informasjonen forskjellig. Fordi vi ikke ønsker å binde feillokaliserings- og omfangsbestemmelsesmetoden til en bestemt informasjonskilde definerer vi her et generelt språk for spesifikasjon av sammensetningen til tjenester, aktiveringssett og komponentsamarbeid. Dette språket danner grunnlaget for utledningene i *del 4*.

3.1 Spesifikasjonsspråket.

Sammensetningen til en enkel tjeneste består her av et aktiveringssett. Vi har to alternativer ved spesifikasjon av sammensetningen til aktiveringssett.

Et aktiveringssett kan beskrives kun ved å angi komponentene som inngår. Dette gir et enkelt spesifikasjonsspråk, men ulempen er at den flate strukturen ikke kan beskrive en hierarkisk oppbygd kommunikasjonsprogramvare tilfredstillende.

Den hierarkiske sammensetningen til kommunikasjonsprogramvare er viktig både ved modellering av årsaksammenhenger mellom komponenter og ved hierarkisk distribusjon av feillokaliserings- og omgangsbestemmelsesmetoden slik vi foreslår i *kapittel 4*. Vi ønsker derfor i tillegg å spesifisere komponentsamarbeid via underliggende tjenester. Da kan den hierarkiske oppbygningen til kommunikasjonsprogramvare spesifiseres rekursivt.

På dette grunnlaget foreslår vi et språk for rekursiv spesifikasjon av sammensetningen til tjenester, aktiveringssett og komponentsamarbeid. Vi benytter en kontekstfri gramatikk til å definere spesifikasjonsspråket.

Kontektsfrie grammatikker.

En kontekstfri grammatikk består av fire komponenter (se [8] for en kort beskrivelse av kontekstfrie grammatikker):

1. Et sett av symboler kalt terminalsymboler.
2. Et sett av ikketerminaler.
3. Et sett av produksjoner som består av en ikketerminal (venstreside), en "->" etterfulgt av en sekvens av terminalsymboler og ikketerminaler.
4. Et startsymbol.

Vi skriver navnet på ikketerminaler med *kursiv ikke-fet* skrift og terminalsymboler med **fet** skrift. For å forenkle grammatikken vil et terminalsymbol med ***kursiv fet*** skrift repre-

sentere en sekvens av bokstaver, siffer og punktum. Terminalsymboler med **ikke-kursiv fet** skrift representerer seg selv. Startsymbolet er venstresiden i første definerte produksjon.

Et språk definert på denne måten består av alle terminalsymbolsekvenser som kan avledes fra startsymbolet. La en *avledning* være en sekvens av terminalsymboler og ikketerminaler. Startsymbolet er initialverdien til avledningen. En terminalsymbolsekvens avledes fra startsymbolet ved å suksessivt erstatte en ikketerminal i avledningen med høyresiden til en produksjon hvis venstreside matcher ikketerminalen.

Spesifikasjonsspråket.

Vi skal nå definere et språk for spesifikasjon av sammensetningen til tjenester, som består av aktiveringssett. Et aktiveringssett består igjen av komponentene som utfører respektiv tjeneste. Komponentene i aktiveringssettet utfører tjenesten ved å enten arbeide selvstendig eller ved å samarbeide via andre underliggende tjenester.

Totalt spesifiseres kommunikasjonssystemet som en liste av tjenester

Kommunikasjonssystem -> Tjenesteliste
Tjenesteliste -> Tjeneste **;** *Tjenesteliste*
Tjenesteliste -> Tjeneste **;**

Hver tjeneste består av et aktiveringssett

Tjeneste -> K **~** *Tjeneste*
Tjeneste -> K

hvor **~** er symbolet for sammenlenking av enkeltkomponenter og samarbeidende komponentpar

K -> ***Komp*** **:** ***Type***
K -> ***Komp*** **:** ***Type*** **(** *Tjeneste* **)** ***Komp***.

Komp angir en komponent, ***Type*** bestemmer komponenttypen samt samarbeidstypen og parentesene representerer en nedstigning i et programvarehierarki.

Strenger i dette språket skal verken skrives eller leses av mennesker, men skal genereres og tolkes automatisk.

3.2 Et eksempel.

Følgende streng spesifiserer sammensetningen til en overføring fra klienten *al-14* over den symbolske forbindelsen *CICS17I* til transaksjonshåndtereren *A01CIC17* (se *figur 1*):

```
CICS17I/al-14:SYM (
   al-14:NL (
         148.119.142.0:SEG~134.47.108.0:SEG
   ) ssi120/IP-del
) CICS17I/ssi120
~
CICS17I/ssi120:SYM (
   TE1L000I-A01CIC17:VL (
         ssi120/SNA-del:NL (
               N120:RING
```

) A01
) A01CIC17-TE1L000I
) A01CIC17;

Strengen danner et forenklet bilde av kommunikasjonsprogramvarehierarkiet. På laveste nivå har vi to typer transmisjonsmedier; segmenter, *SEG*, og ringer, *RING*. "Nettverkslaget", *NL*, representerer alt fra det fysisk nivået til nettverksnivået i en maskin. Over nettverkslaget har vi på SNA-siden et lag som tilbyr virtuelle linjer, *VL*. Hver virtuell linje må defineres både i kildemaskinen og destinasjonsmaskinen slik spesifikasjonen viser. Det øverste laget, *SYM*, tilbyr ende-til-ende kommunikasjon via symbolske forbindelser.

Totalt gir spesifikasjonen av eksempelkommunikasjonssystemet *1494* strenger av denne typen.

3.3 Oppsummering.

Vi har nå definert et språk for spesifikasjon av sammensetningen til tjenester, aktiveringssett og komponentsamarbeid. I *del 4* foreslås hvordan en spesifikasjon i dette språket automatisk kan oversettes til et Bayesnettverk for feillokalisering og omfangsbestemmelse.

4 Feillokalisering og omfangsbestemmelse i kommunikasjonssystemer ved hjelp av et Bayesnettverk.

I denne delen foreslår vi hvordan lokasjonen og omfanget til feil i et kommunikasjonssystem kan bestemmes ved hjelp av et Bayesnettverk. Først under *4.1* velger vi generell fremgangsmåte for feillokalisering og omfangsbestemmelse. Deretter, under *4.2*, begrunnes hvorfor vi har valgt å benytte et Bayesnettverk til oppgaven. Videre, under *4.3*, drøftes Bayesnettverkets generelle struktur samt hvordan strenger i spesifikasjonsspråket fra *del 3* automatisk kan oversettes til et egnet Bayesnettverk. Til slutt argumenteres det for at denne fremgangsmåten gir adaptiv feillokalisering og omfangsbestemmelse med hensyn på den strukturelle sammensetningen til et kommunikasjonssystem. Vi avslutter *del 4* med en oppsummering i *4.4*.

4.1 Tjenesteorientert vs. komponentorientert feillokalisering og omfangsbestemmelse.

Tilstanden i et kommunikasjonssystem kan bestemmest på to måter; *tjenesteorientert* eller *komponentorientert*. Med tjenesteorientert menes at tilstanden i kommunikasjonssystemet bestemmes ved å måle og analysere egenskaper ved tjenestene som tilbys, for eksempel tilgjengeligheten til en overføringstjeneste på nettlagsnivå. Med komponentorientert menes at tilstanden i kommunikasjonssystemet bestemmes ved å måle og analysere komponentegenskaper, for eksempel antall detekterte rammefeil i nettlaget til en ruter. Begge fremgangsmåtene er viktige og kan selvfølgelig kombineres.

Vi ser tre fordeler ved en tjenesteorientert fremfor en komponentorientert fremgangsmåte. For det første får man et direkte og detaljert bilde av tilstanden til tjenestene som tilbys av kommunikasjonssystemet. Dette er den eneste måten å forsikre seg om at tjenestene har ønskede egenskaper; målinger av egenskapene til en komponent kan sjelden utelukke feil i komponenten. La oss for eksempel se på overføringstjenesten mellom klient *al-14* og transaksjonshåndterer *A01CIC17* som ble spesifisert i *del 3*. Selv om man her måler et stort antall egenskaper ved gateway *ssi120*, kan man ikke være sikker på at overføringstjenesten fra klient *al-14* til transaksjonshåndterer *A01CIC17* fungerer. Dette fordi kompleksiteten i gatewayen er så stor at man vanskelig kan foreta et uttømmende sett av målinger.

Den andre fordelen er at man dekker over en rekke komponenter kun ved *en* måling. Ved for eksempel å måle egenskapene til overføringstjenesten fra klient *al-14* til transaksjonshåndterer *A01CIC17*, måles samtidig tilstanden til komponentene i benyttet aktiveringssett. Ved en komponentorientert fremgangsmåte må man utføre målinger i hver eneste komponent.

Den siste fordelen er at man kan lokalisere feil uten å ha direkte tilgang til de underliggende komponentene. Dette er heldig i komplekse heterogene kommunikasjonssystemer, og i kommunikasjonssystemer hvor man av sikkerhetsmessige eller organisasjonsmessige grunner ikke har tilgang til alle komponentene. I Telenors gatewaysystem har man for eksempel ikke tilgang til komponentene i SNA-nettverket fra TCP/IP-nettverket og motsatt. Likevel skal vi se at man ved en tjenesteorientert fremgangsmåte kan lokalisere feil i SNA-nettverket fra klientene i TCP/IP-nettverket.

Fordelen med en komponentorientert fremgangsmåte er rett og slett at man får et direkte og detaljert bilde av tilstanden i komponentene.

Fordi en tjenesteorientert fremgangsmåte har klare fordeler i komplekse og sammensatte kommunikasjonssystemer samt fordi komponentorienterte fremgangsmåter er dekket relativt godt, velger vi her en tjenesteorientert fremgangsmåte.

Neste trinn vil være å inkorporere de to fremgangsmåtene. Dette skisseres nærmere i videre arbeid. I denne delen foreslår vi hvordan man kan oppnå tjenesteorientert feillokalisering og omfangsbestemmelse ved hjelp av et Bayesnettverk.

4.2 Bayesnettverk vs. andre metoder.

Målet er å bestemme lokasjonen og omfanget til feil ved å finne de mest sannsynlige tilstandene i kommunikasjonssystemet gitt kjente tilstander. Det er flere metoder som kan benyttes til denne oppgaven: feed-forward nevrale nettverk [7], radial basis nettverk [7], nærmeste-nabo metoder [7], første ordens predikatlogikk [4], fuzzy logic [9], Bayesnettverk [3], etc.

Metoder som feed-forward nevrale nettverk, radial basis nettverk og nærmeste-nabo metoder kan brukes til å approksimere en direkte avbildning fra for eksempel tjenestetilstander til komponenttilstander. Da bør ideelt sett alle kombinasjoner av tjenestetilstander avbildes. Dette kan være vanskelig fordi antall kombinasjoner vokser eksponensielt med antall tjenester. Videre er en avbildning av denne typen utilstrekkelig hvis man ønsker å inkorporere kjente tilstander både i domenet og funksjonsrommet til avbildningen (f. eks. ved å ta hensyn til kjente komponenttilstander), bortforklare tilstander, eller forklare tilstander. Dette fordi årsaksammenhengene i kommunikasjonssystemet ikke er modellert.

Årsaksammenhengene i et kommunikasjonssystem er preget av usikkerhet. Dermed er en deterministisk årsaksmodell basert på første ordens predikatlogikk utilstrekkelig. Vi står da igjen med fuzzy logic og Bayesnettverk.

Som vi så i *del 1* bør årsaksammenhenger modelleres og håndteres globalt i kommunikasjonssystemet. Fuzzy logic behandler usikkerhet *ekstensjonelt*, dvs at usikkerheten rundt en konklusjon beregnes som en funksjon av usikkerheten rundt premissene. Dermed håndteres ikke usikre årsakssammenhenger globalt (se [3]). Et Bayesnettverk håndterer derimot usikre årsaksammenhenger globalt, dvs *intensjonelt*. Ulempen er at beregningene kan bli uhåndterlige ved store årsaksmodeller.

Ut fra disse betraktningene velger vi å benytte et Bayesnettverk til feillokalisering og omfangsbestemmelse. Dette innebærer en grundig undersøkelse av beregnbarheten til resulterende årsaksmodeller. Vi foretar en slik undersøkelse i *del 5*.

4.3 Bayesnettverket.

Vi utleder her en metode for oversetting av strenger i spesifikasjonsspråket fra *del 3* til et Bayesnettverk som modellerer årsaksammenhenger mellom tilstanden til tjenester, aktiveringssett, komponentsamarbeid og komponenter. Legg merke til at for å forenkle utledningen tar vi kun for oss tilfeller hvor sammensetningen til tjenester er statisk bestemt. I *del 7* skisserer vi hvordan dynamisk bestemte sammensetninger kan håndteres.

Først drøftes to generelle oppbygningsvalg som definerer rammen for Bayesnettverket.

Hukommelsefri vs. hukommelse.

Bayesnettverket kan være med eller uten hukommelse.

Hvis Bayesnettverket er uten hukommelse bestemmes tilstanden i kommunikasjonssystemet direkte fra siste sett av måleresultat. Da er det naturlig å anta at kommunikasjonssystemet fungerte ved forrige sett av måleresultat. Dermed mister man informasjon. Fordelen er at forenklingen gjør fremgangsmåten mindre krevende beregningsmessig.

Den andre muligheten er å følge tilstanden i kommunikasjonssystemet over tid. Dette kan gjøres ved å beregne sannsynligheten for nye *tilstandskonfigurasjoner* fra forrige beregningsresultat, i tillegg til siste sett av målinger. Med tilstandskonfigurasjon menes her en instansiering av variable i et Bayesnettverk. På denne måten får Bayesnettverket hukommelse. Fordi tilstandene sjelden kan bestemmes med full sikkerhet må beregningene utføres for hver av tilstandskonfigurasjonene fra forrige beregning og midles over sannsynligheten til hver av dem (en generell beskrivelse av denne typen dynamiske Bayesnettverk gis i [4]). Fremgangsmåten er derfor beregningsmessig mer krevende enn førstnevnte fremgangsmåte.

Vi antar at feil rettes relativt raskt etter lokalisering. Da vil antagelsen om at kommunikasjonssystemet fungerte ved forrige sett av målinger ikke føre til et for stort tap av informasjon. Videre er gjerne usikkerheten ved målingene så liten at det ofte er unødvendig å samordne dem over tid. Etter som beregningene er flaskehalsen ved metoden er det dermed naturlig å velge førstnevnte fremgangsmåte.

Dette innebærer at kun et øyeblikksbilde av årsaksammenhengene modelleres. Spørsmålet er hvor kompleks årsaksmodellen bør være.

Enkle vs. komplekse årsakssammenhenger.

I et Bayesnettverk modelleres årsakssammenhenger mellom tilstandene til variable. Årsakssammenhengene forenkles ved at en variabel og variablens direkte årsakskilder modelleres uavhengig av de resterende variablene. På denne måten tar man bare hensyn til et subsett av variablene av gangen ved modellering.

Vi ønsker å modellere årsaksammenhengene mellom tilstanden til tjenester, aktiveringssett, komponentsamarbeid og komponenter i et kommunikasjonssystem. Problemet er å avgjøre hvordan årsakssammenhengene skal bygges opp. Komplekse årsakssammenhenger har stor uttrykkskraft men gir krevende beregninger. Enkle årsakssammenhenger gir enkle beregninger men har liten uttrykkskraft.

I den enklest tenkelige modellen vil tilstanden til hver komponentvariabel påvirke tilstanden til tjenestesvariable uavhengig av tilstanden til andre komponentvariable. Dermed kan man ikke uttrykke samarbeid mellom komponentene i kommunikasjonssystemet. La for eksempel to identiske komponenter samarbeide om en gitt tjeneste. Anta videre at tjenesten utføres tilfredstillende så lenge en av komponentene fungerer tilfredstillende. Dette vil ikke kunne uttrykkes i den enklest tenkelige modellen.

I den mest komplekst tenkelige modellen vil enhver komponentvariabel påvirke tilstanden til tjenestesvariable på en måte som er direkte avhengig av tilstanden til alle andre

komponentvariable. Med direkte menes at avhengighetene ikke kan deduseres fra avhengighetskjeder. Da kan man uttrykke svært komplekse sammenhenger, men både modelleringen og beregninger i modellen blir svært krevende.

Vi antar at en så kompleks modell ikke er nødvendig for et kommunikasjonssystem. Dette fordi komponenter gjerne samarbeider parvis via en gitt tjeneste. Dermed kan årsakssammenhenger deduseres ved å lenke sammen slike samarbeid. En overføringstjeneste kan for eksempel bestå av et sett ruterkomponenter som parvis samarbeider slik at data overføres fra ende til ende. Det betyr at hvert samarbeid kan modelleres uavhengig av de andre samarbeidene og deretter lenkes sammen.

I følgende avsnitt foreslår vi et Bayesnettverk som gjenspeiler disse betraktningene ved å gi spesifikasjonsspråket fra *del 3* et semantisk innhold.

Bayesnettverk.

Vi starter med å gi en generell beskrivelse av egenskapene til et Bayesnettverk. Det kan være oppklarende å følge *figur 2* og *figur 3* under denne beskrivelsen. Den interesserte leser kan finne en grundig innføring til Bayesnettverk i [3] og en introduksjon til Bayesnettverk i [5].

Et Bayesnettverk har en kvalitativ og en kvantitativ del. Den kvalitative delen består av en rettet asyklisk graf. En node i grafen representerer en variabel med et gitt endelig antall tilstander. Hver variabel er per definisjon alltid i *en* og bare *en* tilstand. En kant representerer retningen på et direkte årsak-virkning forhold mellom variable. Ved hjelp av disse konstruksjonene kan årsaksammenhenger mellom variable beskrives og følges kvalitativt.

Den kvantitative delen består av en ubetinget eller betinget sannsynlighetsfordeling for hver variabel. Variable uten foreldre i grafen har en ubetinget sannsynlighetsfordeling som representerer *a priori* kunnskap om hvor sannsynlig hver tilstand er. Vi betegner a priori sannsynlighetsfordelingen til en variabel med navn A på forkortet form ved

$$\mathbf{P}(A).$$

Spesifisert fullt ut gir dette en sannsynlighet, $\mathbf{P}(A = a)$, for hver av tilstandene til A. Disse sannsynlighetene skal summere til *1*.

En betinget sannsynlighetsfordeling angir et årsak-virkningforhold numerisk og betegnes på forkortet form ved

$$\mathbf{P}(A \mid B_1, ..., B_n)$$

hvor A er en variabel med foreldrevariablene $B_1, ..., B_n$ i grafen. Skrevet fullt ut angir dette sannsynligheten for en tilstand a gitt tilstand $b_1, ..., b_n$, $\mathbf{P}(A = a \mid B_1 = b_1, ..., B_n = b_n)$, for alle kombinasjoner av tilstander. Her skal sannsynligheten for tilstandene i A gitt en tilstandskonfigurasjon $b_1, ..., b_n$ summere til *1*.

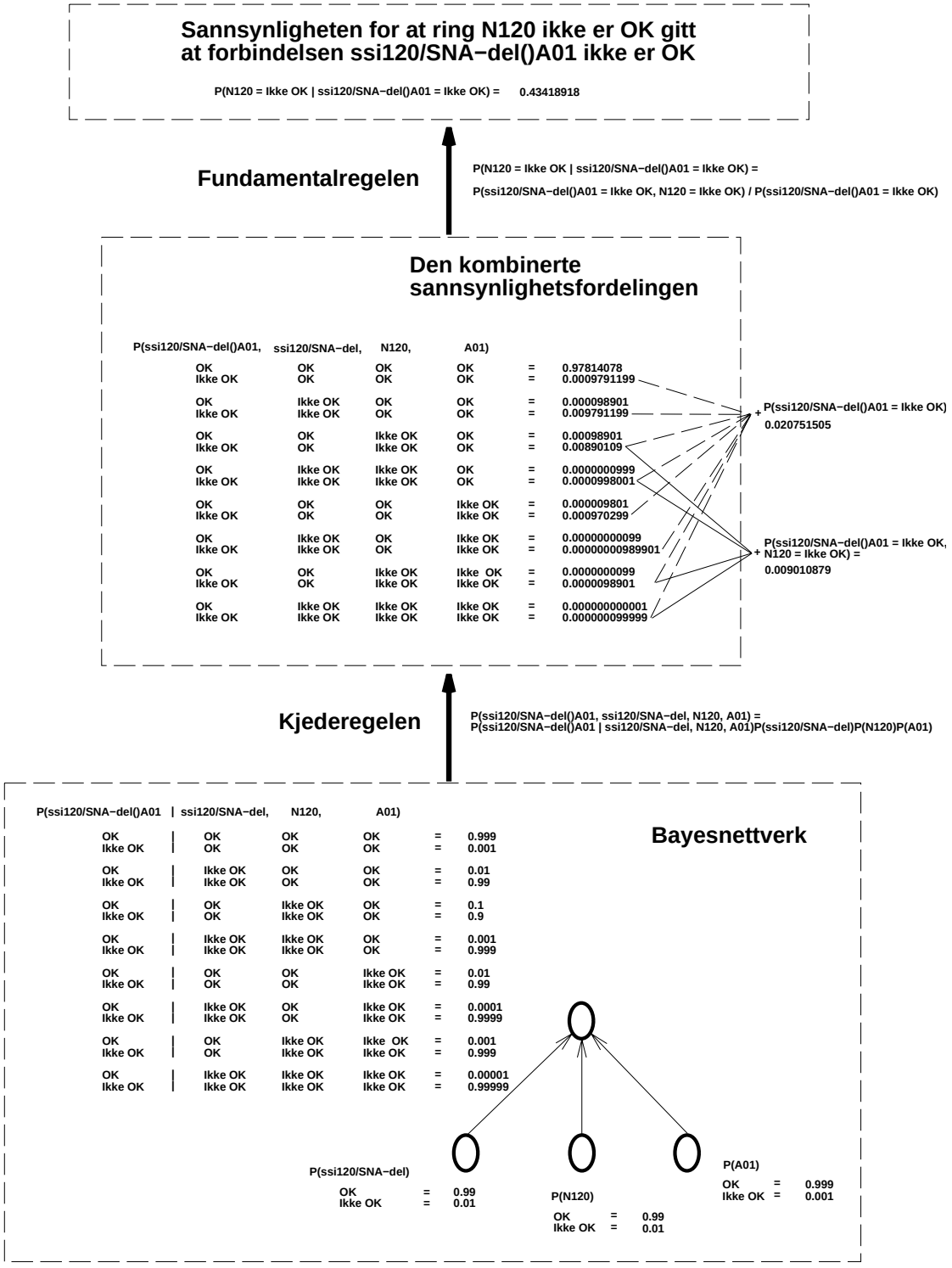
Hvis gitte krav er innfridd kan sannsynligheten til en *global tilstandskonfigurasjon*, dvs en tilstandskonfigurasjon som omfatter alle variablene i Bayesnettverket, beregnes ved å multiplisere sammen sannsynligheten for tilstandskonfigurasjonen i hver av de ubetingede og betingede sannsynlighetsfordelingene. Denne regelen kalles *kjederegelen*. Sannsynlighetsfordelingen over alle globale tilstandskonfigurasjoner kaller vi den *kombinerte sannsynlighetsfordelingen* til Bayesnettverket.

Sannsynligheten for en gitt tilstandskonfigurasjon, $\mathbf{P}(B_1 = b_1, ..., B_p = b_p)$, kan finnes ved å summere sannsynligheten til alle globale tilstandskonfigurasjoner hvor gitt tilstandskonfigurasjon finner sted. Videre kan *fundamentalregelen* benyttes til å beregne

sannsynligheten for enhver variabeltilstand $A = a$ gitt kjente variabeltilstander $B_1 = b_1$, ..., $B_p = b_p$

$$P(A = a | B_1 = b_1, ..., B_p = b_p) = \frac{P(A = a, B_1 = b_1, ..., B_p = b_p)}{P(B_1 = b_1, ..., B_p = b_p)}$$

Se *figur 2* for et eksempel på beskrevne beregninger.



Figur 2: *Et komplett Bayesnettverk med et beregningseksempel.*

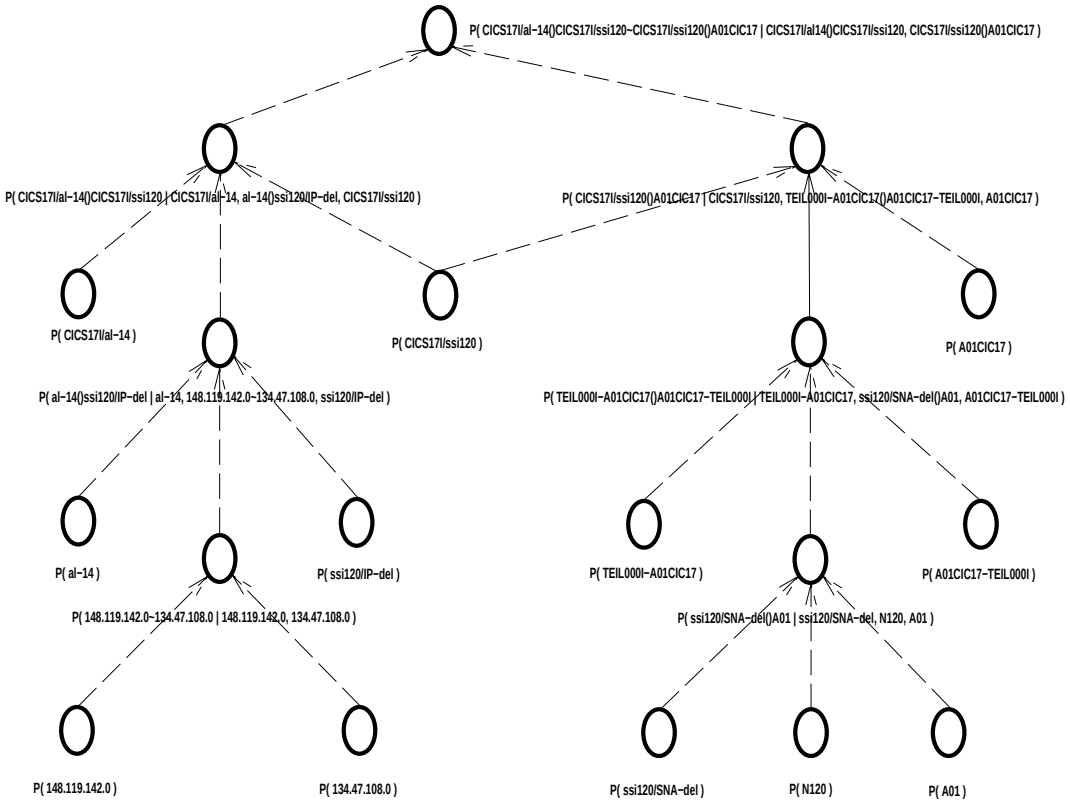
Litt forenklet håndterer fremgangsmåten årsaksammenhenger globalt (intensjonelt) fordi alle operasjoner foregår på et globalt plan, dvs på den kombinerte sannsynlighetsfordelingen. Problemet er at antall sannsynligheter i den kombinerte sannsynlighetsfordelingen vokser eksponensielt med antall variable slik at direkte beregninger raskt blir uhåndterlige. Vi ser på ulike beregningsmetoder som omgår dette problemet i *del 5*. Her foreslår vi hvordan strenger i spesifikasjonsspråket fra *del 3* kan oversettes til et Bayesnettverk som modellerer årsakssammenhengene mellom tilstanden til tjenester, aktiveringssett, komponentsamarbeid og komponenter.

Fra spesifikasjon til Bayesnettverk - et eksempel.

Vi bygger opp Bayesnettverket ved å sette sammen instanser av predefinerte ubetingede og betingede sannsynlighetsfordelinger. Hvilke sannsynlighetsfordelinger som settes sammen og hvordan de settes sammen bestemmes av spesifikasjonen. Som vi så i *del 3* har en streng i spesifikasjonspråket en rekursiv struktur. En streng kan da oversettes ved hjelp av et *oversettelsesskjema*. Før vi definerer et egnet oversettelseskjema ser vi på hvordan eksempelstrengen

```
CICS17I/al-14:SYM (
   al-14:NL (
         148.119.142.0:SEG~134.47.108.0:SEG
   ) ssi120/IP-del
) CICS17I/ssi120
~
CICS17I/ssi120:SYM (
   TE1L000I-A01CIC17:VL (
         ssi120/SNA-del:NL (
               N120:RING
         ) A01
   ) A01CIC17-TE1L000I
) A01CIC17;
```

kan oversettes til Bayesnettverket i *figur 3*.



Figur 3: *Et Bayesnettverk som modellerer årsaksammenhenger fra eksempelstrengen.*

Bayesnettverket i *figur 3* modellerer *en* tjeneste. I det fullstendige eksempel-Bayesnettverket er *1494* slike del-Bayesnettverk flettet sammen.

Vi tar først for oss modelleringen av komponenter, deretter komponentsamarbeid og til slutt aktiveringssett.

Modellering av komponenter.

I Bayesnettverket modellerer vi en komponent som en variabel (node). Vi lar tilstandene til variabelen være abstrakte komponenttilstander som viser hvilke *komplette* tilstander en komponenten kan ha. Med komplett menes hele den interne tilstanden i ett. Tilstandene bør være på et abstraksjonsnivå som fanger essensen i problemstillingen uten å forkludre modellen med detaljer. Ved feillokalisering og omfangsbestemmelse kan for eksempel en binær tilstand være tilstrekkelig (*OK* eller *ikke OK*).

Vi lar komponentvariablene være uten foreldre i Bayesnettverket. Det betyr at vi ser på komponentene i et kommunikasjonssystem som uavhengige utgangspunkt for feil. I videre arbeid ønsker vi også å undersøke hvordan direkte årsaksammenhenger mellom tilstanden til komponenter kan modelleres, og hvilken innvirkning slike årsakssammenhenger vil ha på metodens nøyaktighet og effektivitet.

I eksempelstrengen har vi følgende komponenter:

CICS17I/al-14, al-14, 148.119.142.0, 134.47.108.0, ssi120/IP-del, CICS17I/ssi120, TE1L000I-A01CIC17, ssi120/SNA-del, N120, A01, A01CIC17-TE1L000I, A01CIC17.

Fordi vi ikke ønsker å spesifisere a priori sannsynlighetsfordelingen til hver eneste komponentvariabel for seg, deler vi komponentene opp i typer og gir hver type en sannsynlighetsfordeling. Sannsynlighetsfordelingen til en komponentvariabel blir dermed en instans

av typesannsynlighetsfordelingen. Komponentvariabel *148.119.142.0* er for eksempel av type *SEG* og får da denne typens sannsynlighetsfordeling. Vi betegner dette ved

$$\mathbf{P_{SEG}(}\textit{148.119.142.0}\mathbf{)}.$$

Se *figur 2* for et eksempel på slike sannsynlighetsfordelinger.

Modellering av komponentsamarbeid.

Vi går nå over til å modellere komponentsamarbeid. Et komponentsamarbeid modelleres ved å sette sammen to komponentvariable og en tjenestevariabel. Vi lar navnet på samarbeidsvariabelen være ***Komp*()*Komp'*** hvor ***Komp*** og ***Komp'*** er navnet på hver av de samarbeidende komponentvariablene. Videre lar vi tilstanden til samarbeidsvariabelen være direkte avhengig av tilstanden til komponentvariablene og tjenestesvariabelen, ***tjeneste***. Det innføres dermed en rettet kant fra ***Komp***, ***tjeneste*** og ***Komp'*** til ***Komp*()*Komp'*** i Bayesnettverket.

I eksempelstrengen får samarbeidet mellom gateway *ssi120/SNA-del* og MVS *A01* navnet *ssi120/SNA-del()A01*. Dette samarbeidet foregår via overføringstjenesten som tilbys av ring *N120*. Tilstanden til overføringstjenesten *ssi120/SNA-del()A01* er altså avhengig av tilstanden til *ssi120/SNA-del*, *N120* og *A01*. Vi får dermed en rettet kant fra hver av de tre komponentvariablene til samarbeidsvariabelen *ssi120/SNA-del()A01* i Bayesnettverket (se *figur 3*).

Samarbeidstypen angis ved typen til komponentene, men med suffikset *rel*. I eksempelet er komponenttypen *NL*. Samarbeidstypen er da *NLrel*. For hver samarbeidstype definerer vi en betinget sannsynlighetsfordeling. Sannsynlighetsfordelingen spesifiserer årsaksforholdet mellom tilstanden til to samarbeidende komponentvariable, tjenestesvariabelen og tilstanden til samarbeidsvariabelen, numerisk. I eksempelet får vi den betingede sannsynlighetsfordelingen

$$\mathbf{P_{NLrel}(}\textit{ssi120/SNA-del()A01} \mid \textit{si120/SNA-del, N120, A01}\mathbf{)}.$$

Se *figur 2* for et eksempel på en slik sannsynlighetsfordeling.

Modellering av aktiveringssett.

Vi går nå over til å modellere aktiveringssett. Et aktiveringssett modelleres ved en kjede av "fuzzy" og-relasjoner. På denne måten kan vilkårlig store aktiveringssett modelleres. Vi betegner og-sannsynlighetsfordelingen med

$$\mathbf{P_{og}(}\textit{og-kjede} \sim \textit{K} \mid \textit{og-kjede}, \textit{K}\mathbf{)}$$

hvor ***og-kjede*** representerer navnet på et eksisterende aktiveringssett og ***K*** representerer navnet på en komponentvariabel eller en komponentsamarbeidsvariabel. Et aktiveringssett *148.119.142.0~134.47.108.0* modelleres dermed ved

$$\mathbf{P_{og}(}\textit{148.119.142.0~134.47.108.0} \mid \textit{148.119.142.0, 134.47.108.0}\mathbf{)}.$$

Oversettelsesskjemaet.

Vi har nå skissert hvordan Bayesnettverket i *figur 3* kan bygges fra eksempelstrengen, nedenfra og opp. Her vises hvordan en streng som spesifiserer sammensetningen til tjenester kan oversettes til et Bayesnettverk ved hjelp av et oversettelsesskjema.

Et oversettelsesskjema er et skjema som knytter oversettelsesregler til produksjonene i en kontekstfri grammatikk. Her gis en forenklet beskrivelse av funksjonen til et oversettelsesskjema slik vi bruker det. I [8] gis en mer grundig innføring.

Et oversettelsesskjema evalueres over *avledningstreet* man får ved avledningen av en streng. Et avledningstre viser hvordan en streng avledes fra startsymbolet og har følgende egenskaper:

1. Startsymbolet danner roten i treet.
2. Hver bladnode angir et terminalsymbol eller ε. ε angir her en tom høyreside i en produksjon.
3. Hver intern node angir en ikketerminal.
4. Hvis *A* er en ikketerminal i en intern node og X_1, X_2, ..., X_n er barna til noden, så er *A* -> $X_1X_2...X_n$ en produksjon i grammatikken. Her står *A* for en ikketerminal og X_1, X_2, ..., X_n for terminal- eller ikketerminalsymboler. Ved produksjonen *A* -> ε kan en node *A* ha ett barn ε.

Et oversettelsesskjema slik vi bruker det, evalueres ved å utføre en dybdeførst traversering fra venstre mot høyre i avledningsstreet til en gitt streng. Etter traverseringen av barna til en node evalueres regelen som hører til "avledningsproduksjonen", dvs produksjonen som avleder barna fra gitt node. Evalueringsresultat lagres i noden slik at at de blir gjort tilgjengelige for foreldrenoden.

Vi ønsker å generere tjenestesvariablene i Bayesnettverket nedenfra og opp. Det gjør vi ved å la navnet på tjenestesvariable propagere oppover i avledningstreet. På denne måten kan stadig mer sammensatte tjeneste defineres fra underliggende tjenestesvariable. Oversettelsesskjemaet vi benytter defineres i *tabell 1*.

Produksjon	**Semantiske regler**
Kommunikasjonssystem -> Tjenesteliste	
Tjenesteliste -> Tjeneste **;** *Tjenesteliste*	
Tjenesteliste -> Tjeneste **;**	
K -> ***Komp*** **:** ***Type***	**P**$_{Type}$**(*Komp*)**; *K* := ***Komp***;
K -> ***Komp*** **:** ***Type*** **(** *Tjeneste* **)** ***Komp'***	**P**$_{Type}$**(*Komp*)**; **P**$_{Type}$**(*Komp'*)**; **P**$_{Typerel}$**(*Komp*()*Komp'* | *Komp*,** *Tjeneste*, ***Komp'*)**; *K* := ***Komp*** **()** ***Komp'***;
Tjeneste -> K **~** *Tjeneste'*	**P**$_{og}$**(***K*~*Tjeneste'* | *Tjeneste'*, *K***)**; *Tjeneste* := *K* ~ *Tjeneste'*;
Tjeneste -> K	*Tjeneste* := *K*;

Tabell 1: *Et oversettelsesskjema for oversetting av strenger i spesifikasjonsspråket til et Bayesnettverk for feillokalisering og omfangsbestemmelse.*

I oversettelsesskjemaet lagres et evalueringsresultat i node *A* ved operasjonen *A := evalueringssresultat*. Evalueringsresultatet til et nodebarn aksesseres ved navnet på barnet.

En sannsynlighetsfordeling til en variabel *A* i Bayesnettverket defineres ved notasjonen $\mathbf{P}_{\textbf{\textit{Fordeling}}}(A)$ hvis *A* er uten foreldre og $\mathbf{P}_{\textbf{\textit{Fordeling}}}(A \mid B_1, ..., B_n)$ hvis *A* har $B_1, ..., B_n$ som foreldre i Bayesnettverket. ***Fordeling*** angir navnet på en gitt sannsynlighetsfordeling. For å forenkle skjemaet kan variable defineres flere ganger. Da er det alltid den siste definisjonen som gjelder.

4.4 Konklusjon.

Foreslåtte Bayesnettverk modellerer årsakssammenhenger mellom tilstanden til tjenester, aktiveringssett, komponentsamarbeid og komponenter globalt i kommunikasjonssystemet. Videre håndterer beregninger i Bayesnettverk årsakssammenhenger globalt. På denne måten har vi oppnådd global feillokalisering og omfangsbestemmelse (**H1**).

En spesifikasjon i det definerte spesifikasjonsspråket kan genereres automatisk fra kommunikasjonssystem eller fra konfigurasjons- og topologidatabaser. Videre kan Bayesnettverket genereres automatisk fra en spesifikasjon i språket ved hjelp av det definerte oversettelsesskjemaet. Forsøk i eksempelkommunikasjonssystemet viser at oversettelsen er svært rask, så rask at Bayesnettverket kan generes fra bunn av ved endringer i den strukturelle sammensetningen til et kommunikasjonssystem. Vi konkluderer dermed med at resulterende feillokaliserings- og omfangsbestemmelsesmetode er adaptiv med hensyn på sammensetningen til et kommunikasjonssystem (**H2**).

I neste del undersøker vi egenskapene til foreslått feillokaliserings- og omfangsbestemmelsesmetode ved å undersøke nøyaktigheten og effektiviteten til ulike beregningsmetoder i Bayesnettverk generert fra eksempelkommunikasjonssystemet.

5 Beregninger i modell.

Lokasjonen og omfanget til feil bestemmes i foreslåtte Bayesnettverk ved å beregne de mest sannsynlige tilstandene i kommunikasjonssystemet fra kjente tilstander. Generelt er slike beregninger uhåndterlige, men en rekke beregningsmetoder gir gode resultater for mange Bayesnettverk.

Først under *5.1* ser vi på noen tradisjonelle beregningsmetoder. Deretter under *5.2* foreslår vi modifikasjoner av disse for bruk i våre Bayesnettverk. Beskrevne beregningsmetoder sammenlignes så under *5.3* ved et eksperiment basert på en modell av eksempelkommunikasjonssystemet fra *del 2*. Til slutt under *5.4* konkluderer vi med en oppsummering av resultatene fra denne delen.

5.1 Tradisjonelle beregningsmetoder.

Det er to hovedfremgangsmåter for beregning av sannsynlighetsfordelingen til variablene i et Bayesnettverk gitt kjente tilstander. Beregningene kan enten gjøres nøyaktig ved å summere over hele den kombinerte sannsynlighetsfordelingen gitt kjente tilstander eller approksimativt ved å summere over en delmengde av den kombinerte sannsynlighetsfordelingen til Bayesnettverket gitt kjente tilstander (se *del 4* for en kort beskrivelse av Bayesnettverk, den kombinerte sannsynlighetsfordelingen til et Bayesnettverk og beregning av sannsynlighetsfordelingen til en variabel ved bruk av summering og fundamentalregelen). Vi skisserer her først egenskapene til tradisjonelle metoder for nøyaktige beregninger. Deretter skisserer vi egenskapene til tradisjonelle metoder for approksimative beregninger.

Nøyaktige beregningsmetoder.

Hvis Bayesnettverket gir et tre når retningen på kantene ikke tas i betraktning, kan beregningene utføres nøyaktig i tid lineært på antall variable (se [3]). Bayesnettverkene i dette kapittelet har dessverre ikke denne egenskapen.

Vi tar for oss to hovedberegningsmetoder for generelle Bayesnettverk. Den ene metoden, *The Method of Conditioning* [3], går ut på å instansiere variabler til beregningene kan utføres i en rekke uavhengige trær. Problemet er at antall trær vokser eksponensielt med antall instansieringer. Den andre metoden, *Clustering Methods* [3], danner et tre av variabelklynger slik at variable som danner sykler i Bayesnettverket (når man ikke tar hensyn til retningen på kantene) slås sammen i en eller flere klynger. Problemet er at størrelsen på en klynge vokser eksponensielt med antall variable i klyngen.

Vi startet undersøkelsene med sistnevnte fremgangsmåte. Beregningsmetoden ga hurtige og nøyaktige beregninger i innledningsfasen da vi jobbet med mindre Bayesnettverk. Dessverre ble størrelsen på klyngene uhåndterlige da vi utvidet detaljnivået og størrelsen på Bayesnettverkene. Dette førte til at vi gikk over til approksimative beregningsmetoder.

Approksimative beregningsmetoder.

Approksimative beregningsmetoder håndterer lettere store og komplekse Bayesnettverk fordi man kun ønsker å finne de mest sannsynlige tilstandskonfigurasjonene i den kombinerte sannsynlighetsfordelingen, gitt kjente tilstander. De viktigste approksimative beregningsmetodene beskrives i [6]. Vi har undersøkt de tre metodene som skisseres her.

Den første metoden, *Maximum Probability Search Method* [6], gjør et deterministisk søk i årsaksretningen etter de mest sannsynlige tilstandskonfigurasjonene i Bayesnettverket gitt kjente tilstander. Søket utføres som følger:

1. Variablene i Bayesnettverket sorteres topologisk.
2. For hver av tilstandene til første variabel i sorteringen (en variabel uten foreldre i Bayesnettverket) dannes en rotnode i et søketre. En rotnode representerer variabelen i en gitt tilstand. Hver node gis en verdi lik den ubetingede sannsynligheten til tilstanden noden representerer. Dette danner utgangspunktet for søket etter de mest sannsynlige tilstandskonfigurasjonene i den kombinerte sannsynlighetsfordelingen til Bayesnettverket gitt kjente tilstander.
3. Bladnoden med høyest verdi identifiseres. Denne bladnoden representerer en variabel i en gitt tilstand. Etterfølgervariabelen til denne variabelen i sorteringen identifiseres. For hver av de mulige tilstandene (kun en hvis tilstanden er kjent) til etterfølgervariabelen utvides den identifiserte bladnoden med ett barn. Dette barnet representerer etterfølgervariabelen i en gitt tilstand. Hvert barn gis en verdi lik verdien til foreldrenoden multiplisert med en sannsynlighet fra sannsynlighetsfordelingen til etterfølgervariabelen. Denne sannsynligheten er unikt bestemt av tilstandene representert av barnet og nodene i grenen.
4. Hvis en bladnode representerer siste variabel i den topologiske sorteringen har vi funnet en tilstandskonfigurasjon i den kombinerte sannsynlighetsfordelingen til Bayesnettverket. Denne tilstandskonfigurasjonen er mer sannsynlig enn hver av de gjenstående tilstandskonfigurasjonene i søketrærne og har sannsynlighet lik verdien til bladnoden. Grenen som representerer denne tilstandskonfigurasjonen kan fjernes.
5. Hopp til *3.* hvis sannsynligheten til flere tilstandskonfigurasjoner er ønsket.
6. Normaliser til slutt sannsynligheten til identifiserte tilstandskonfigurasjoner, dvs sørg for at sannsynlighetene summerer til en.

Fordelen ved dette søket er at sannsynligheten til tilstandskonfigurasjonene i den kombinerte sannsynlighetsfordelingen identifiseres etter fallende sannsynlighet. Dermed er man alltid sikker på at gjenstående tilstandskonfigurasjoner er mindre sannsynlig en de man allerede har funnet. Ulempen ved denne fremgangsmåten er litt forenklet at søketreet vokser eksponensielt med antall forgjengere hvis mest sannsynlige tilstander er i konflikt med kjente tilstander til etterfølgerene. Da må søket gjennom alle kombinasjoner av mer sannsynlige deltilstandskonfigurasjoner før den finner en deltilstandskonfigurasjon som "passer" de kjente tilstandene. Dette gir klare utslag i eksperimentet som beskrives under *5.3*; ved store Bayesnettverk, ved støy eller ved samtidige feil blir søketrærne uhåndterlige før metoden kommer frem til en tilstandskonfigurasjon. Først etter en ad-hoc modifikasjon ble denne metoden tilfredstillende til vår bruk. Vi skisserer denne modifikasjonen under *5.2*.

Ulempen ved beskrevet deterministisk søk i årsaksretningen er som nevnt at søketrærne kan bli uhåndterlig store. Uhåndterlig store søketrær kan byttes ut med uhåndterlig lang søketid ved å foreta et stokastisk søk i årsaksretningen (*Acceptance-Rejection Sampling* eller *forward sampling* [6]). Med et stokastisk søk i årsaksretningen menes her følgende metode:

1. Variablene i Bayesnettverket sorteres topologisk.
2. Søket itererer over variablene i sortert rekkefølge.
3. Hvis en besøkt variabel ikke har foreldre i Bayesnettverket trekkes tilstanden til variabelen fra variabelens ubetingede sannsynlighetsfordeling. Hvis variabelen har foreldre identifiseres tilstanden til disse (allerede bestemt). Variabelens betingede sannsynlighetsfordeling og tilstanden til foreldrene benyttes så til å trekke tilstanden til variabelen (se [6] for detaljer).
4. Hvis en trukket tilstand kommer i konflikt med kjente tilstander forkastes trukket tilstandskonfigurasjon og søket begynner på nytt fra første variabel i den topologiske sorteringen.
5. Hvis søket treffer de kjente tilstandene slik at variabelene i Bayesnettverket er i en uregistrert tilstandskonfigurasjon, kan tilstandskonfigurasjonen registreres.
6. Når ikke flere tilstandskonfigurasjoner ønskes, normaliseres til slutt sannsynligheten til identifiserte tilstandskonfigurasjoner.

Den normaliserte sannsynlighetsfordelingen til tilstandskonfigurasjonene generert ved denne metoden konvergerer til den kombinerte sannsynlighetsfordelingen spesifisert i Bayesnettverket gitt kjente tilstander. Ulempen er at når de kjente tilstandene er lite sannsynlige gitt de mest sannsynlige tilstandene til forgjengervariablene vil denne fremgangsmåten ta uhåndterlig lang tid (de fleste tilstandskonfigurasjonene må forkastes). Vi har foreløpig sett bort fra denne metoden da initielle forsøk med stokastiske søk både i og mot årsaksretningen var lite vellykkede.

Den tredje metoden gjør et stokastisk søk ved å la Bayesnettverket være i en tilstandskonfigurasjon som endres under søket. Metoden finner de mest sannsynlige tilstandskonfigurasjonene gitt kjente tilstander som følger:

1. Variablene i Bayesnettverket gis en initiell tilstand (gjerne en tilfeldig). Variable med kjent tilstand låses i denne. Deretter besøkes variablene i en eller annen rekkefølge.

2. Når en variabel besøkes benyttes tilstanden til nabovariablene (foreldrene, barna og barnas foreldre) og variabelens og barnas sannsynlighetsfordeling til å trekke tilstanden til variabelen (se [7] for detaljer). Dermed påvirkes tilstanden til variabelen både i og mot årsaksretningen.
3. Når Bayesnettverket er i en ny uregistrert tilstandskonfigurasjon registreres denne tilstandskonfigurasjonen.
4. Når ikke flere tilstandskonfigurasjoner ønskes, normaliseres til slutt sannsynligheten til identifiserte tilstandskonfigurasjoner.

Den normaliserte sannsynlighetsfordelingen til tilstandskonfigurasjonene generert ved denne metoden konvergerer til den kombinerte sannsynlighetsfordelingen spesifisert i Bayesnettverket gitt kjente tilstander.

En slik sampler kalles en *Gibbs-sampler* (se [7] for en beskrivelse). Fordelen ved en Gibbs-sampler er at den alltid genererer tilstandskonfigurasjoner (et deterministisk eller stokastisk søk i årsaksretningen behøver ikke komme frem til noen løsning hvis de kjente tilstandene er lite sannsynlige gitt de mest sannsynlige tilstandene til forgjengervariablene). Videre vil effekten av kjente tilstander stegvis propagere gjennom Bayesnettverket. Dermed vil alle kjente tilstander uavhengig av a priori sannsynlighet påvirke genererte sampler. Ulempen med denne metoden er at nabovariablene kan tvinge hverandre i globalt suboptimale tilstandskonfigurasjoner. I eksperimentet kom denne ulempen klart frem; i situasjoner med alternative feilkilder låste metoden seg ved flere anledninger i en av de alternative tilstandskonfigurasjonene. Dermed ble det tilfeldig om korrekte feilkilder ble lokalisert. I enkelte feilsituasjoner, spesielt i de hvor måleresultatene var i konflikt med hverandre på grunn av støy, låste metoden seg i tilstandskonfigurasjoner som var svært usannsynlige globalt. Dermed genererte metoden for oss ubrukelige tilstandskonfigurasjoner.

Ut fra initielle forsøk er ingen av beregningsmetodene tilfredstillende i vår sammenheng. Vi foreslår derfor under neste punkt noen modifikasjoner hvor målet er å omgå ulempene ved de ulike metodene. Metodene sammenlignes under *5.3* ved et eksperiment.

5.2 Egne modifikasjoner.

Her foreslår vi to modifikasjoner. Den første modifikasjonen er en ad-hoc endring av søkekriteriet i det deterministiske søket. Den andre modifikasjonen er et forslag til en Gibbs-sampler som hindrer lokal låsing ved å åpne direkte "kanaler" til kjente tilstander i Bayesnettverket.

Modifisert deterministisk søk.

Ved det deterministiske søket får man verdien til en bladnode i en gren ved å multiplisere de betingede og ubetingede sannsynlighetene gitt av deltilstandskonfigurasjonen grenen representerer. Dette medfører at en bladnode i en gren gjerne får lavere verdi dess lenger grenen er. Dermed får søket en svekket fullføringsevne. Anta at vi har et kommunikasjonssystem hvor sannsynligheten for at en komponent fungerer er *0.995* og at sannsynligheten for at et komponentsamarbeid fungerer når komponentene fungerer er *0.995*. Anta videre at korresponderende Bayesnettverk har *2000* komponentvariable og samarbeidsvariable. Anta til slutt at alle kommunikasjonssystemtjenestene modellert i Bayesnettverket fungerer. I en så enkel situasjon vil likevel den svekkede fullføringsevnen gi et stort søketre. Den lengste grenen før fullførelse av søket etter mest sannsynlige tilstandskonfigurasjon vil ha verdien

$P(Komponent/Samarbeid = OK)^{1999} = 0.0000445.$

Det betyr litt forenklet at andre grener i søketreet ekspanderes så lenge bladnodene i grenene har høyere verdi enn $P(Komponent/Samarbeid = OK)^{1999}$. Dermed blir søketrærne store selv i enkle situasjoner. Ved samtidige feil eller støy har vi erfart at søketrærne blir uhåndterlige.

Vi foreslår i stedet å bruke midlet sannsynlighet som bladnodeverdi under søket (men selvsagt ikke ved beregning av sannsynligheten til resulterende tilstandskonfigurasjoner). Da vil fullføringsevnen til søket styrkes:

$P(Komponent/Samarbeid = OK) * 1999 / 1999 = 0.995.$

Man er riktignok ikke garantert å finne den mest sannsynlige tilstandskonfigurasjonen først med dette søkekriteriet, men eksperimentet under *5.3* viser at denne ad-hoc modifikasjonen gir bedre søk i våre Bayesnettverk.

Modifisert Gibbs-sampler.

I en Gibbs-sampler kan man hindre at variable låses i lokalt sannsynlige men globalt usannsynlige tilstander ved å kombinere flere variable (en blokk) ved trekking av tilstander. Kort beskrevet beregner man den kombinerte sannsynlighetsfordelingen til variablene innenfor en blokk nøyaktig og trekker så en tilstandskonfigurasjon fra denne sannsynlighetsfordelingen. Dermed åpnes direkte "kanaler" mellom variabeltilstandene i blokken.

Denne metoden er relativt ny og det finnes for øyeblikket få retningslinjer for gruppering av variable [7]. I vår situasjon er valg av blokker spesielt vanskelig fordi Bayesnettverkene genereres dynamisk. Videre kan man ikke se bort fra at den kombinerte sannsynlighetsfordelingene innenfor "nyttige" blokker blir uhåndterlig store. Vi foreslår derfor her en alternativ form for trekking av tilstander i tillegg til en dynamisk gruppering i blokker.

Metoden fungerer som følger:

1. Variablene i Bayesnettverket gis en tilfeldig tilstandskonfigurasjon. Variable med kjent tilstand låses i denne.
2. Det itereres over variablene uten foreldre, dvs komponentvariablene i våre Bayesnettverk.
3. Når en variabel *A* besøkes utføres følgende operasjoner:

a) Etterfølgerene til *A* i Bayesnettverket identifiseres, dvs alle tjenestevariable som bygger på *A* i vår sammenheng. Disse variablene grupperes i en blokk. På denne måten åpnes direkte "kanaler" til kjente tjenestetilstander som kan være forårsaket av *A*.

Hvis en tilstandskonfigurasjon trekkes fra den kombinerte sannsynlighetsfordelingen i blokken oppnås to fordeler relativt til en tradisjonell Gibbs-sampler. Nabovariable vil ikke tvinge hverandre inn i lokalt sannsynlige, men globalt usannsynlige tilstandskonfigurasjoner. Sampleren vil ikke låse seg i en av flere alternative forklaringer. Dette kan begrunnes litt forenklet som følger:

Anta at vi har en blokk som inneholder kjente tilstander. Enten forklares alle de kjente tilstandene av variable utenfor blokken eller så forklares ikke alle de kjente tilstandene av variable utenfor blokken.

Hvis alle de kjente tilstandene ikke forklares av variable utenfor blokken, vil tilstanden til de andre variablene i blokken trekkes fra en sannsynlighetsfordeling betinget på de

uforklarte kjente tilstandene. Dermed finner metoden raskt en tilstandskonfigurasjon som forklarer de kjente tilstanden i Bayesnettverket.

Hvis alle de kjente tilstandene forklares vil tilstanden til de andre variablene i blokken trekkes fra a priori sannsynlighetsfordelingen, dvs sannsynlighetsfordelingen når kjente tilstander ikke influerer tilstanden til variablene. Hvis sannsynlighetsfordelingen innenfor en blokk ikke inneholder ekstreme sannsynligheter, innebærer dette at variablene i blokken relativt raskt vil forklare kjente tilstander i blokken ved en tilfeldighet. Dermed får vi to forklaringer for en eller flere av de kjente tilstandene. Ved neste iterasjon vil den andre forklaringen (i en annen blokk) sannsynligvis falle bort.

På denne måten unngås at sampleren låses i lokalt sannsynlige tilstandskonfigurasjoner eller i en av flere alternative forklaringer.

Den kombinerte sannsynlighetsfordelingen i beskrevne blokker vil i store og komplekse Bayesnettverk bli uhåndterlig store. Vi trekker derfor ikke tilstandskonfigurasjoner direkte fra den kombinerte sannsynlighetsfordelingen. I stedet trekker vi kun tilstanden til *A* fra sannsynlighetsfordelingen til *A* nøyaktig beregnet innenfor blokken betinget på tilstanden til variable utenfor blokken. Denne sannsynlighetfordelingen kan beregnes med en av metodene beskrevet under punkt *5.1*. I eksempelkommunikasjonssystemet danner variablene i foreslåtte blokker trær (hvis man fjerner det overflødige symbolske abstraksjonsnivået i TCP/IP-nettverket). Det betyr at sannsynlighetsfordelingen kan beregnes i lineær tid med hensyn på antall variable i blokken.

b) Vi lar tilstanden til *A* forplante seg til *A* sine etterfølgere ved å utføre *3.* i hvert av barna. På denne måten slipper man å beregne den kombinerte sannsynlighetsfordelingen.

5.3 Eksperimentresultat.

Under dette punktet sammenligner vi først de ulike beregningsmetodene med to eksperimenter egnet til dette. Deretter undersøker vi hvordan den best egnede beregningsmetoden håndterer økende grad av støy og et stigende antall samtidige feil.

Eksperimentene.

I eksperimentene genereres tilfeldige feilsituasjoner i eksempelkommunikasjonssystemet fra *del 2*. Dette innebærer kunstig bestemmelse av tilstanden til de symbolske forbindelsene i kommunikasjonssystemet gitt tilstanden til komponentene. De kunstig bestemte tilstandene mates til hver av beregningsmetodene. Resultatet fra hver beregningsmetode er sannsynligheten for ulike feilkilder. Disse resultatene benyttes først til å sammenligne beregningsmetodene, og deretter til å vurdere feillokaliserings- og omfangsbestemmelsesmetodens anvendbarhet, dvs hvor nøyaktig og effektiv metoden er.

Problemet er å bestemme tilstanden til de symbolske forbindelsene kunstig. Vi har gjort følgende erfaringer med det virkelige systemet

- ❑ En komponent feiler som regel totalt. Det betyr at alle symbolske forbindelser avhengig av komponenten også feiler.
- ❑ Ved feil i en komponent øker sannsynligheten for at symbolske forbindelser som ikke er direkte avhengig av komponenten også feiler. Denne type feil oppstod i 6 av 57 observerte feilsituasjoner.
- ❑ Det er mulig at en komponent ikke feiler totalt. Det vil si at en symbolsk forbindelse som er avhengig av komponenten fungerer med en viss sannsynlighet.

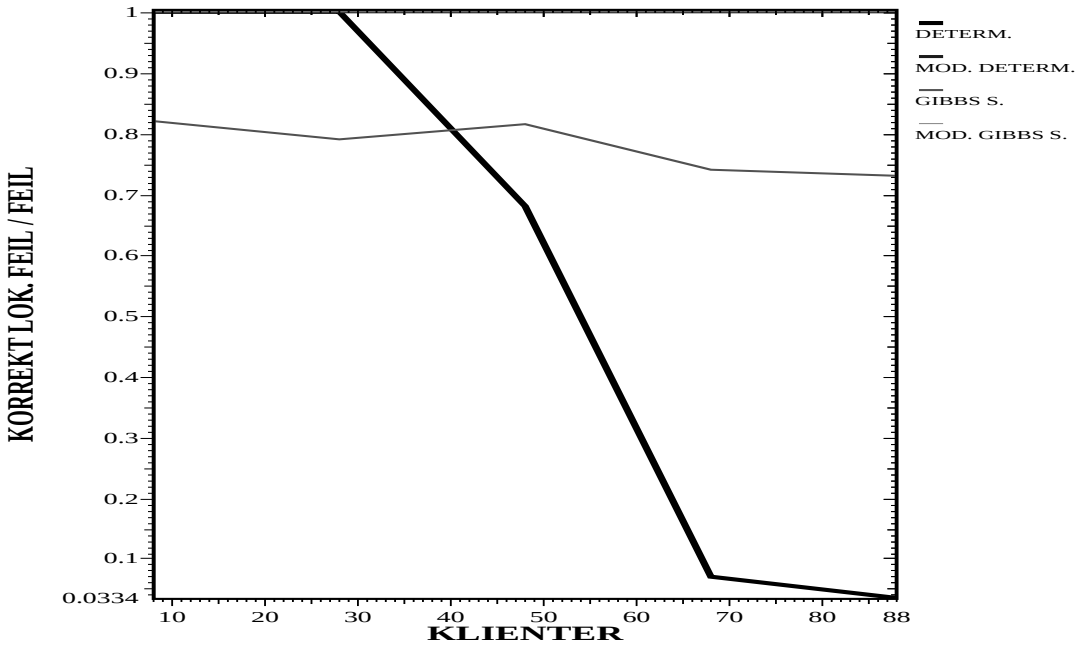
Her følger en detaljert punktvis beskrivelse av hvordan eksperimentene er konstruert.

- Hver av de *1504* forbindelsene knyttes til komponentene som inngår i forbindelsen.
- Vi genererer en feilsituasjon med *n* samtidige feil ved å trekke *n* komponenter tilfeldig. Trekkingen er basert på feilraten til komponentene. Vi gir komponentene en feilrate på *0.001* bortsett fra gatewayene og ringene som vi gir en feilrate på *0.01*. Dette fordi feil oppstår oftere i ringene og gatewayene enn i de andre komponentene og fordi feil i disse gir de mest interessante feilsituasjonene.
- Når feilede komponenter er trukket identifiseres forbindelsene hvor disse komponentene inngår. Disse forbindelsene er *direkte berørt* av feilsituasjonen.
- Ved totale feil feiler alle direkte berørte forbindelser. Ved ikke-totale feil trekkes hvilke direkte berørte forbindelser som feiler. Vi gir forbindelser direkte berørt av en feilet komponent en feilsannsynlighet på *0.999* bortsett fra forbindelser direkte berørt av feilede ringer. Disse forbindelsene gir vi feilsannsynlighet på *0.95*. Dette for å representere at ringene erfaringsmessig har en større tilbøyelighet til å feile delvis.
- I en virkelig feilsituasjon kan andre enn de berørte forbindelsene også feile. Dette på grunn av økt belastning på kommunikasjonssystemet eller på grunn av for oss ukjente årsakssammenhenger. Vi simulerer denne usikkerheten ved å la uberørte forbindelser feile med en gitt sannsynlighet. Denne sannsynligheten settes til ulike verdier avhengig av eksperimentene.
- For å kunne justere størrelsen på resulterende Bayesnettverk grupperes forbindelsene etter klientene sortert i alfabetisk rekkefølge. Det minste kommunikasjonssystemet består av de *9* første klientene i sorteringen, forbindelsene som opprettes av klientene samt underliggende komponenter. Vi øker størrelsen på kommunikasjonssystemet ved å stegvis innføre *20* klienter med respektive forbindelser og underliggende komponenter. Det minste Bayesnettverket blir da på *500* variable mens det største (*89* klienter) blir på *2467* variable.
- Det genereres på beskrevet måte *200* feilsituasjoner for hver unik konfigurasjon med hensyn på antall samtidige feil, graden av totale/ikke-total feil, graden av støy, størrelsen på kommunikasjonssystemet og beregningsmetode.
- Hver beregningsmetode kjører i *20* sekunder på en *SUN, Sparc Ultra-1* maskin. Hvis en beregningsmetode fungerer tilfredstillende innenfor denne tidsrammen regner vi beregningsmetoden som effektiv.
- Vi måler to egenskaper i eksperimentene. Den ene egenskapen er *treffgrad,* det andre egenskapen er *nøyaktighet*. Med treffgrad menes antall feilede komponenter hvor beregningsmetoden estimerte en feilsannsynlighet høyere enn *0.1* dividert på antall feilede komponenter (*antall korrekt lokaliserte feil/antall feil*). Hvis det for eksempel genereres *100* feil og benyttet beregningsmetode gir *90* av disse en feilsannsynlighet høyere enn *0.1* er treffgraden *0.9*. Med nøyaktighet menes summen av estimerte sannsynligheter for feil høyere enn *0.1* hvor komponentene ikke har feilet, dividert på summen av estimerte sannsynligheter for feil høyere enn *0.1* hvor komponentene har feilet (*uriktig vektlegging/korrekt vektlegging*). Anta for eksempel at en feilsituasjon har to like sannsynlige alternative forklaringer. Da kan en beregningsmetode for eksempel gi hver av forklaringene en sannsynlighet på *0.4*. Nøyaktigheten blir da *0.4/0.4* som er lik *1*. Da om lag halvparten av genererte feilsituasjoner har *2* eller flere alternative forklaringer forventer vi en nøyaktighet rundt *0.5*.
- Hvis treffgraden er nær *1.0* og nøyaktigheten er nær *0.5* regner vi en beregningsmetode som egnet i gitt eksperimentkonfigurasjon.

For å sammenligne beregningsmetodene utfører vi først to eksperimenter. I de to siste eksperimentene undersøker vi hvordan den best egnede beregningsmetoden håndterer økende grader av støy og et stigende antall samtidige feil.

Resultat fra eksperiment 1.

I første eksperiment består hver feilsituasjon av en feilet komponent. Komponentene feiler totalt og feilsituasjonene er støyfrie. Størrelsen på underliggende kommunikasjonssystem økes med *20* klienter av gangen fra *9* klienter til *89* klienter. Dette gir oss et Bayesnettverk som øker i størrelse fra *500* variable til *2467* variable. Treffgraden til beregningsmetodene under dette eksperimentet illustreres i *plot 1*. Nøyaktigheten illustreres i *plot 2*.

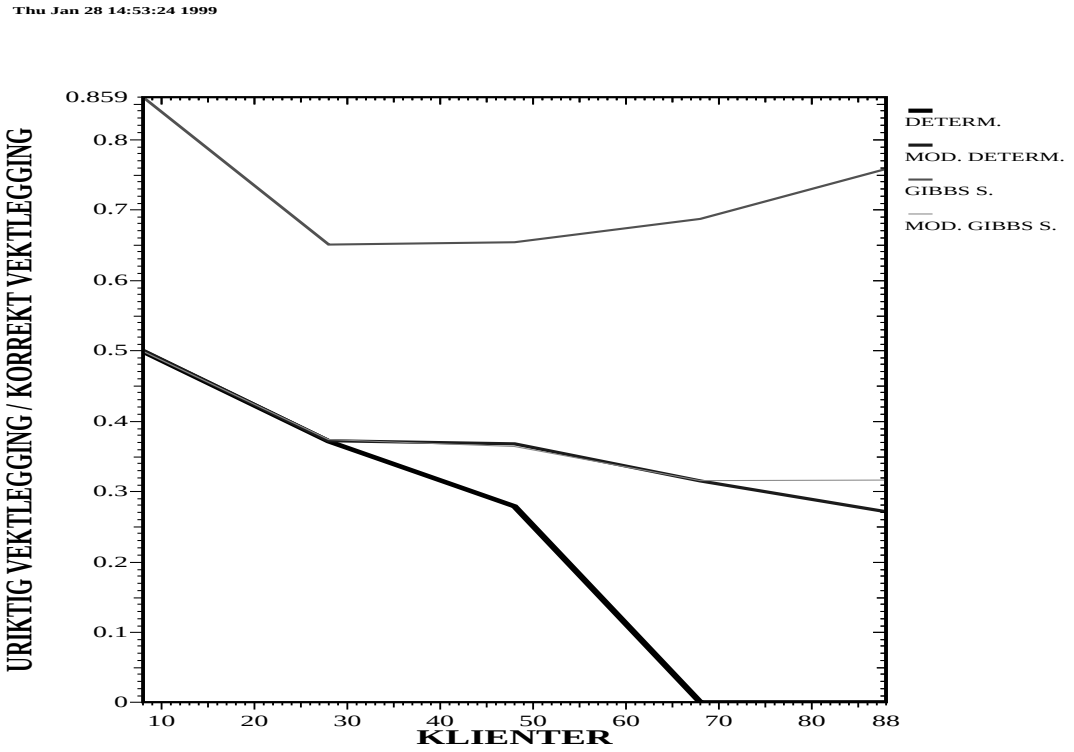


Plot 1: *En totalt feilet komponent - uten støy.*

I *plot 1* ser vi at det deterministiske søket bare håndterer et fåtall klienter (dvs små Bayesnettverk). Dette skyldes at søkekriteriet gir uhåndterlig store søketrær i de store Bayesnettverkene. Det modifiserte deterministiske søket har derimot et søkekriterie med bedre fullføringsevne. Dette gir seg klart utslag i eksperimentet.

Vi ser videre at Gibbs-sampleren har relativt lav treffgrad i forhold til de modifiserte metodene. Dette er fordi Gibbs-sampleren relativt hyppig låser seg i en av flere alternative forklaringer til kjente tilstander. Eksperimentet viser dermed at den modifiserte Gibbs-sampleren ikke låser seg like ofte som den tradisjonelle Gibbs-sampleren.

Det er også viktig å sammenligne hvor nøyaktige beregningsmetodene er. Dette illustreres i *plot 2*.

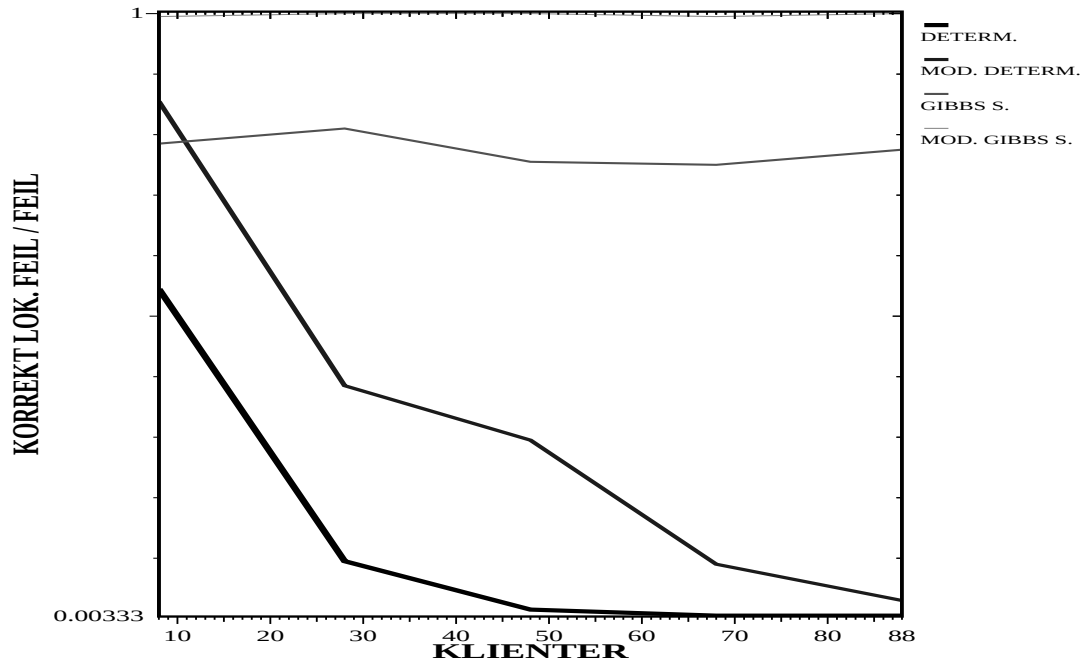


Plot 2: *En totalt feilet komponent - uten støy.*

I *plot 2* ser vi at den modifiserte Gibbs-sampleren og det modifiserte søket er omtrent like nøyaktige. Denne nøyaktigheten er nær den optimale, da om lag halvparten av de genererte feilsituasjonene kunne forklares av to eller flere alternative feilkilder. Den tradisjonelle Gibbs-sampleren er ikke like nøyaktig. Dette fordi den låser seg i suboptimale tilstandskonfigurasjoner. Den optimale nøyaktigheten øker med antall klienter fordi vi får mer informasjon om komponentene i kommunikasjonssystemet når antall forbindelser øker.

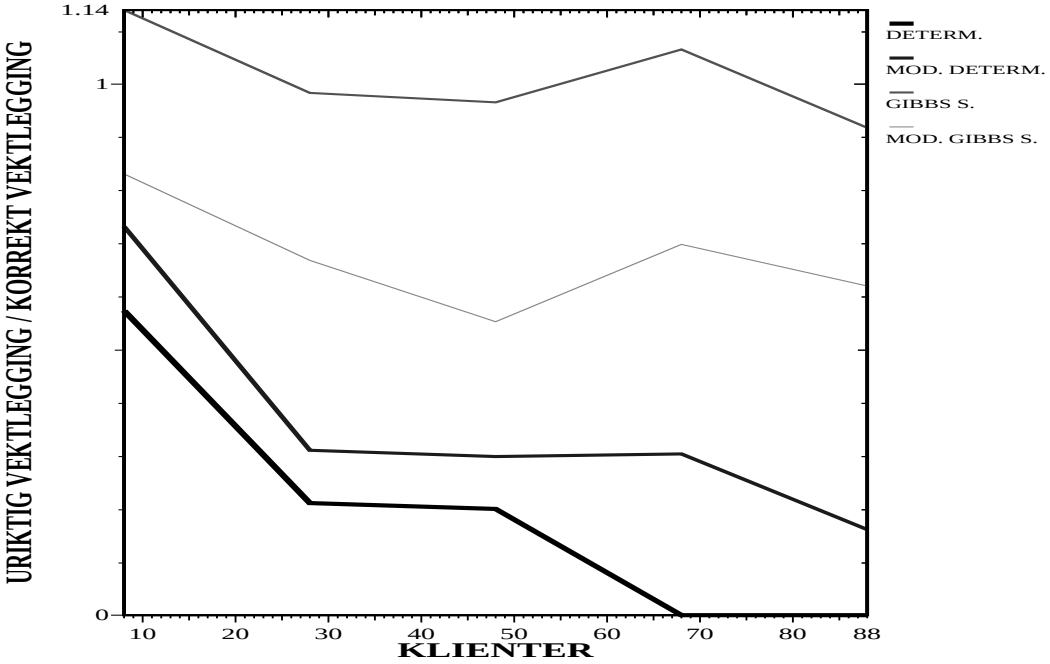
Resultat fra eksperiment 2.

I andre eksperiment består hver feilsituasjon også av en feilet komponent, men her feiler komponentene delvis og feilsituasjonene er preget av støy. Vi lar som nevnt hver direkte berørt forbindelse feile med sannsynlighet 0.999 bortsett fra forbindelser direkte berørt av feilede ringer. Disse forbindelsene feiler med sannsynlighet 0.95. Vi har innført støy ved at andre forbindelser enn de direkte berørte feiler med sannsynlighet 0.005. I det største kommunikasjonssystemet får vi da i snitt 7.5 feilede forbindelser som ikke er direkte berørt av feilede komponenter. Dette er nært støynivået i de mest ekstreme tilfellene vi har erfart i det virkelige kommunikasjonssystemet. Størrelsen på underliggende kommunikasjonssystem varieres også her. Treffgraden til beregningsmetodene under dette eksperimentet illustreres i *plot 3*. Nøyaktigheten illustreres i *plot 4*.



Plot 3: *En delvis feilet komponent - støy.*

I *plot 3* ser vi at begge Gibbs samplerene har relativt upåvirket treffgrad ved innført støy og ikke-totale feil, mens de deterministiske søkene feiler drastisk. Dette skyldes at søke-trærne blir uhåndterlige med gitte søkekriterier.



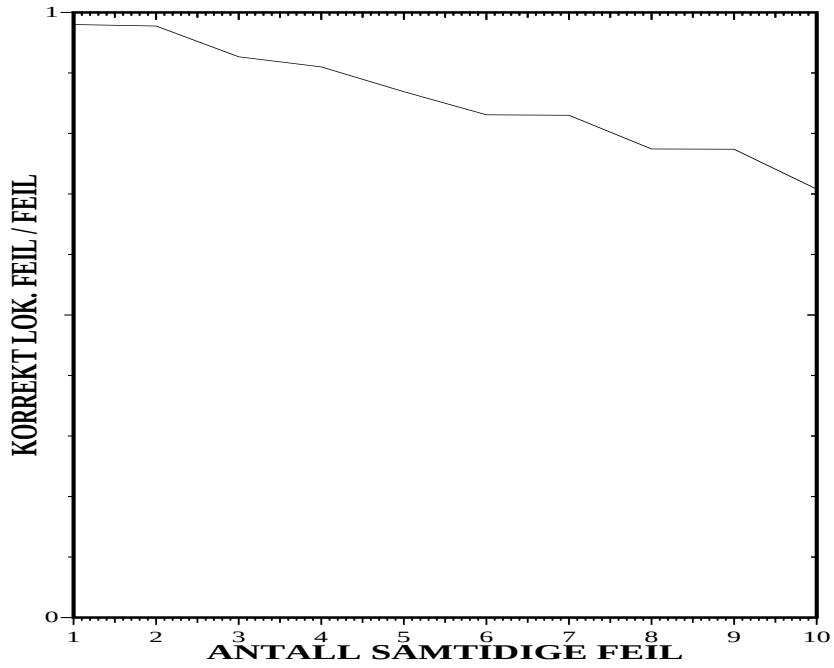
Plot 4: *En delvis feilet komponent - støy.*

Plot 4 viser at ved innført støy og ikke-totale feil er Gibbs-samplerene ikke like nøyaktige, men økningen er ikke drastisk. Økningen skyldes at mange av feillokasjonene kun kan lokaliseres fra tilstanden til en eller to forbindelser. Dermed blir beslutningsgrunnlaget sårbart mot støy. Vi regner derfor den modifiserte Gibbs-sampleren som egnet også ved innført støy og ikke-totale feil.

Resultat fra eksperiment 3.

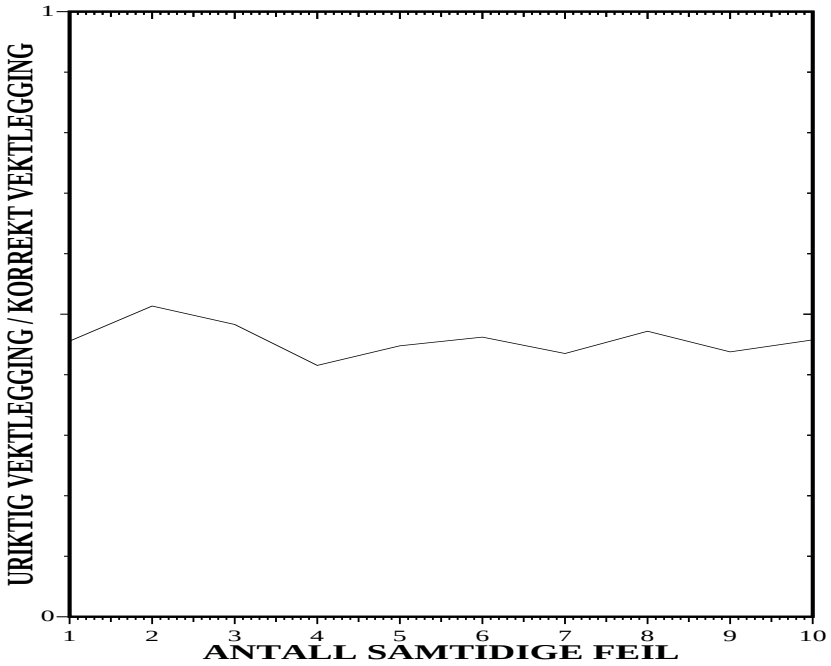
Fra foregående eksperimenter ser vi at den modifiserte Gibbs-sampleren er best egnet til vår bruk. For å få et klarere bilde av begrensningene til den modifiserte Gibbs-sampleren utfører vi to eksperimenter. Først tester vi den modifiserte Gibbs-sampleren med et økende antall samtidige feil. I neste eksperiment tester vi Gibbs-sampleren under stigende grad av støy.

I dette eksperimentet har vi støynivå og ikke-totale feil som i forrige eksperiment. Eksperimentet er basert på det største kommunikasjonssystemet, dvs at alle klientene er med. I tillegg genereres flere samtidige feil. Treffgraden til den modifiserte Gibbs-sampleren under dette eksperimentet illustreres i *plot 5*. Nøyaktigheten illustreres i *plot 6.*



Plot 5: *Flere samtidige feil.*

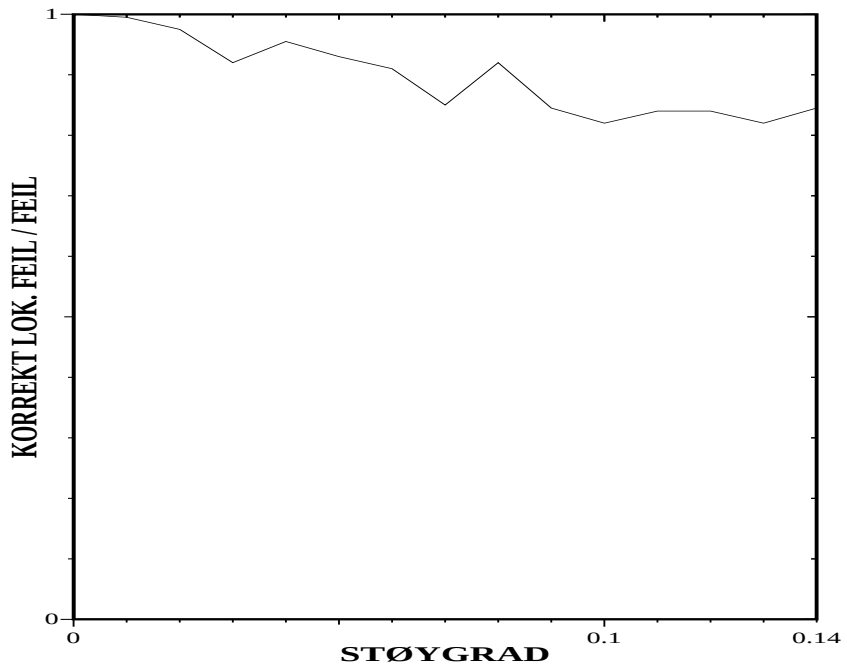
Plottet viser at treffgraden faller med antall samtidige feil. Resultatene fra genererte feilsituasjoner viser at dette skyldes feilsituasjoner hvor en feilet komponent *skygger* over andre feilede komponenter. Hvis for eksempel en ring feiler samtidige med en eller flere av gatewayene knyttet til ringen, er beslutningsgrunnlaget kun tilstrekkelig til å lokalisere den feilede ringen. Den fallende treffgraden skyldes altså i stor grad naturlige begrensninger i beslutningsgrunnlaget og ikke selve beregningsmetoden. Dette kommer klarere frem i *plot 6* hvor vi ser at nøyaktigheten i stor grad er upåvirket av antall samtidige feil.



Plot 6: *Flere samtidige feil.*

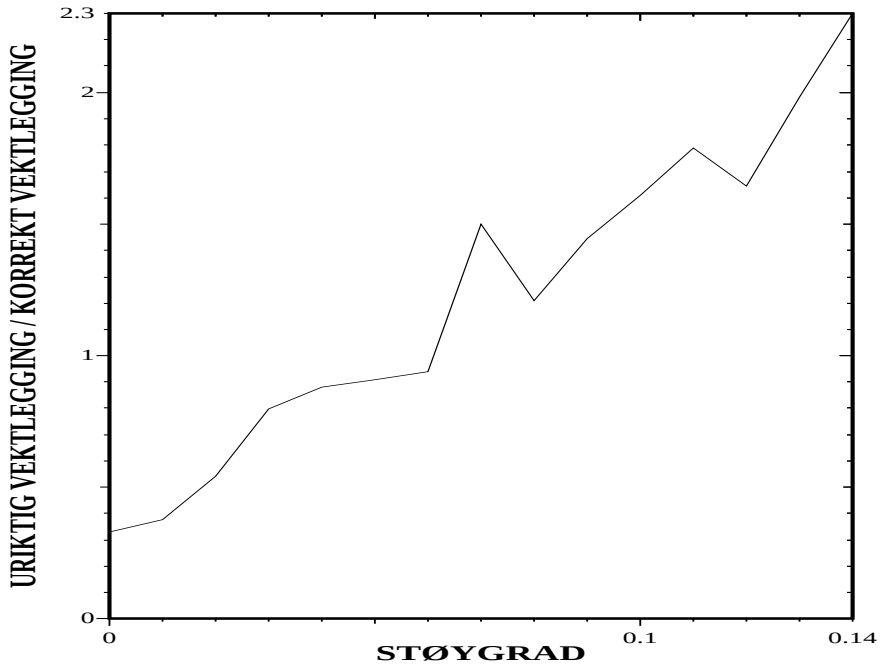
Resultat fra eksperiment 4.

Til slutt undersøker vi hvordan den modifiserte Gibbs-sampleren håndterer støy. Ved innføring av støy tydeliggjøres en grunnleggende svakhet ved beslutningsgrunnlaget; enkelte komponenter inngår bare i en eller to forbindelser. Dette gir følgende dilemma. Enten kan man legge stor vekt på feilede forbindelser slik at selv *en* feilet forbindelse holder til å lokalisere feil. Da vil støy kunne gi et stort utslag. Resultatet av denne fremgangsmåte illustreres i *plot 7* og *plot 8*. I stedet kan man legge mindre vekt på feilede forbindelser slik at for eksempel minst *15* feilede forbindelser entydig må peke ut en feillokasjon. Da vil støy håndteres bedre, men man ofrer mulighetene til å lokalisere feil i lite benyttede komponenter. *Plot 9* og *plot 10* illustrerer denne fremgangsmåten.



Plot 7: *Stigende grad av støy - stor vektlegging av feilede forbindelser.*

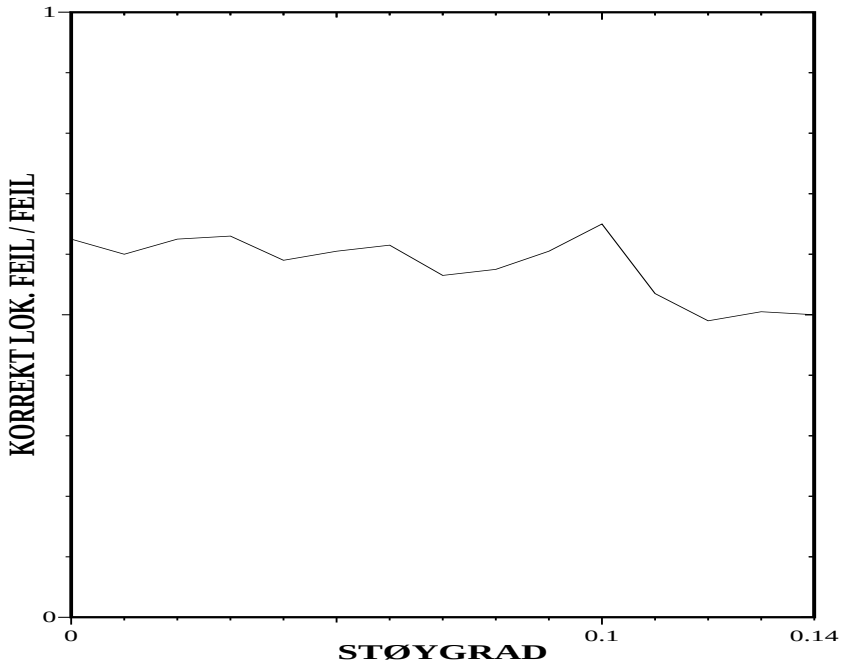
Ved stor vektlegging av feilede forbindelser ser vi i *plot 7* at treffgraden faller svakt ved økende støygrad. I det mest ekstreme tilfellet hvor gjennomsnittlig ca. 210 ikke direkte berørte forbindelser feiler er treffgraden 0.85. Problemet er at feillokaliseringen ikke blir nøyaktig nok ved stor grad av støy. Dette ser vi i *plot 8*.



Plot 8: *Stigende grad av støy - stor vektlegging av feilede forbindelser.*

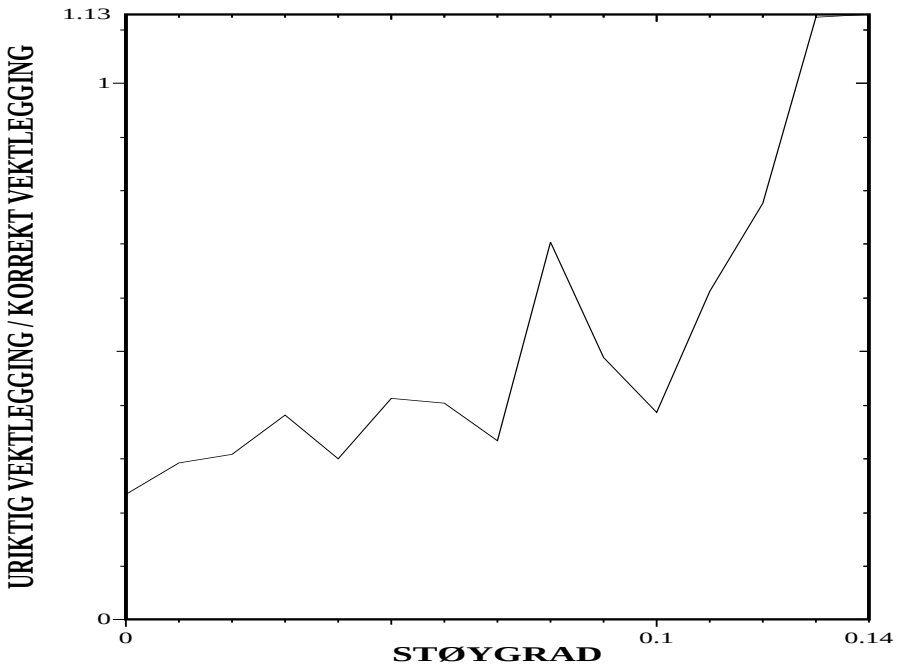
Plottet viser at når sannsynligheten for at en ikke direkte berørt forbindelse feiler er større enn *0.06* blir feillokaliseringen svært unøyaktig. Men dette tilsvarer at gjennomsnittlig ca 90 ikke direkte berørte forbindelser feiler. Dette er langt over det som erfaringsmessig har vært tilfellet i de mest ekstreme tilfellene i det virkelige kommunikasjonssystemet. Da feilet om lag *10* ikke direkte berørte forbindelser.

Det andre alternativer er å legge mindre vekt på feilede forbindelser. Resultatet av denne fremgangsmåten beskrives her.



Plot 9: *Stigende grad av støy - mindre vektlegging av feilede forbindelser.*

Plott 9 viser at treffgraden er lavere enn ved den andre fremgangsmåten. Dette skyldes at en stor del av komponentene inngår kun i en eller to forbindelser. Fordelen ved en mindre vektlegging på feilede forbindelser ser vi i *plot 10*.



Plot 10: *Stigende grad av støy - mindre vektlegging av feilede forbindelser.*

I dette plottet ser vi at nøyaktigheten er akseptabel helt opp til en støygrad på 0.12. Da feiler i snitt om lag *180* av de ikke direkte berørte forbindelsene.

Ulempen med lav treffgrad kan begrenses ved å innføre flere forbindelser i kommunikasjonssystemet eller ved å samordne tilstanden til forbindelsene over tid. Dette er derimot kun nødvendig i kommunikasjonssystemer med svært mye støy.

5.4 Konklusjon.

Vi har drøftet egenskapene ved noen tradisjonelle beregningsmetoder. Disse beregningsmetodene er ikke tilfredstillende i vår sammenheng. Vi har derfor foreslått to modifikasjoner; et nytt søkekriterie for det deterministiske søket og en blokk-basert Gibbs-sampler. Resultatet av eksperimentene viser at den blokk-baserte Gibbs-sampleren er effektiv og nøyaktig selv i feilsituasjoner med ikke-totale feil og med høy grad av støy. Vi har tidligere vist at feillokaliserings- og omfangsbestemmelsesmetoden modellerer og håndterer årsaksammenhenger globalt i kommunikasjonssystemet (**H1**) og at metoden er adaptiv med hensyn på sammensetningen til et kommunikasjonssystem (**H2**). Vi konkluderer dermed med at feillokaliserings- og omfangsbestemmelsesmetoden kan lokalisere og bestemme omfanget til feil nøyaktig og effektivt (**H3**).

I neste del undersøkes feillokaliserings- og omfangsbestemmelsesmetodens praktiske verdi.

6 Implementasjon og praktiske resultat.

I denne delen undersøkes feillokaliserings- og omfangsbestemmelsesmetodens praktiske verdi.

Feillokaliserings- og omfangsbestemmelsesmetoden skal benyttes til å lokalisere feil i Telenors TCP/IP-SNA gatewaysystemer (beskrevet i *del 2*). Kommunikasjonssystemet gir oss et realistisk, heterogent og komplekst miljø hvor vi kan undersøke egenskapene til

feillokaliserings- og omfangsbestemmelsesmetoden. De omfattende og kritiske applikasjonene som kjører i dette kommunikasjonssystemet gjør det videre viktig at feil lokaliseres nøyaktig og effektivt.

Systemekspertene som har ansvaret for de symbolske forbindelsene gjennom gatewayene lokaliserer feil som følger.

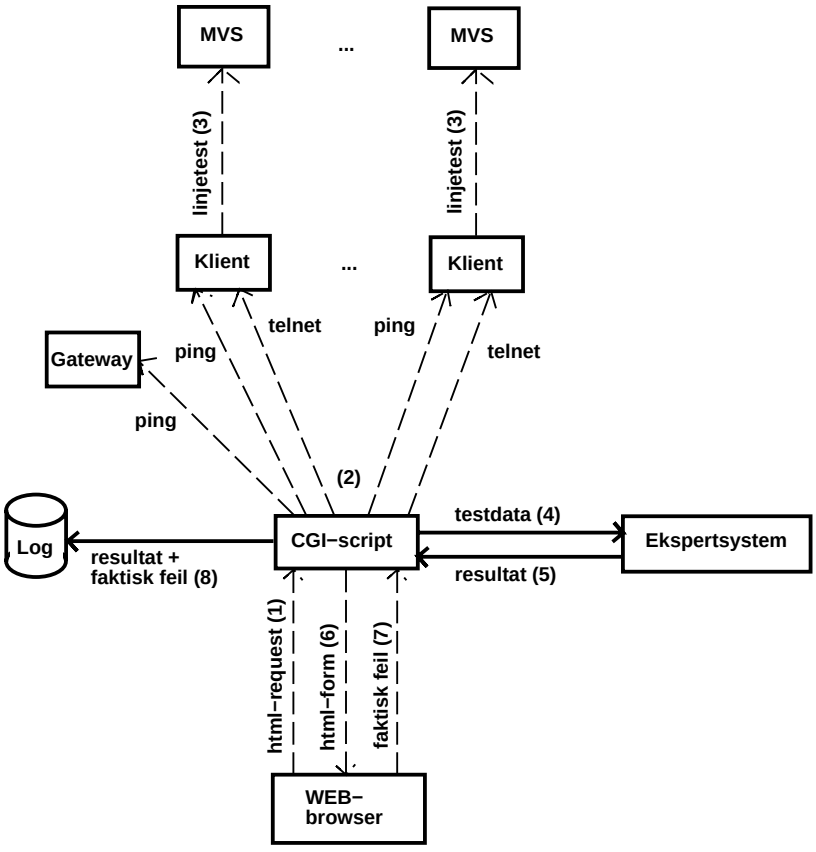
- ❑ Ved å tolke feilmeldinger fra kunder.
- ❑ Ved å logge seg inn på klienter og gatewayer for å teste symbolske forbindelser og lese av komponentmålinger.
- ❑ Ved å føre en dialog med systemeksperter ansvarlig for klienter, TCP/IP-nettverket, SNA-nettverket eller MVSene ved mistanke om feil.
- ❑ Ved å sette feilsymptomer fra overnevnte kilder i sammenheng med den gjeldende konfigurasjonen i kommunikasjonssystemet.

Beskrevet prosess kan i verste fall ta flere timer.

6.1 Implementasjon.

Vi har supplert prosessen med et automatisk, globalt og adaptivt feillokaliserings- og omfangsbestemmelsessystem. Under dette punktet skisseres implementasjonen av feillokaliserings- og omfangsbestemmelsessystemet. *Figur 4* illustrerer systemarkitekturen.

Systemarkitektur.



Figur 4: *Systemarkitektur.*

Hvert av trinnene illustrert i figuren beskrives her før vi tar for oss WWW-grensesnittet og selve ekspertsystemet.

1. Forespørsel om analyse fra WWW-browser aktiverer et *CGI-program*. Med et CGI-program menes et program som behandler HTML-skjemaer.
2. CGI-programmet pinger først gatewayene og klientene i kommunikasjonssystemet. Deretter opprettes det en telnet-sesjon med hver klient.
3. Under telnet-sesjonene henter CGI-programmet ned tilstanden til de overvåkede symbolske forbindelsene.
4. Ping-resultatene og tilstanden til overvåkede symbolske forbindelser overføres til et ekspertsystem.
5. Resultatet av feillokaliseringen utført av ekspertsystemet returneres til CGI-programmet.
6. CGI-programmet genererer et HTML-skjema som presenterer resultatene fra ekspertsystemet.
7. Faktisk feil registreres i HTML-skjemaet. Skjema sendes deretter til CGI-programmet.
8. Feillokaliseringsresultatet + faktiske feil logges i en database (foreløpig manuelt og noe upresist).

Vi tar nå for oss WWW-grensesnittet og tilknyttet ekspertsystem.

WWW-grensesnittet og ekspertsystemet.

Feillokaliseringsystemet aktiveres via et WWW-grensesnitt.

Ved aktivering genererer CGI-programmet en oversikt over feilede pinger og feilede symbolske forbindelser. *Figur 5* illustrerer en slik oversikt.

Feilede tester.

FEILEDE FORBINDELSER.

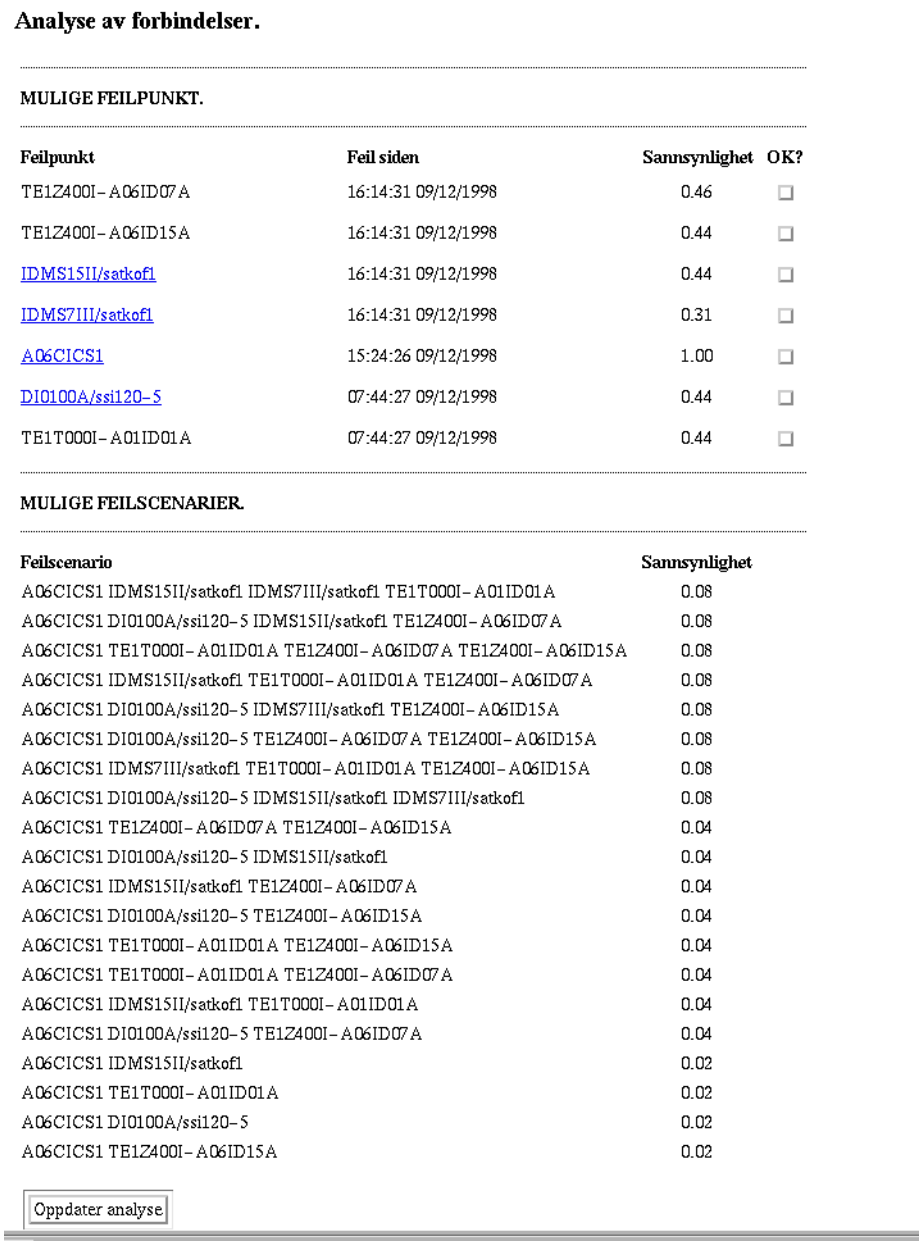
Klient	Symbolic dst	GW	LU	CICS/IDMS
br-rds-3	CICS1I	ssi120-4	TE1T202I	A06CICS1
br-rds-3	DC0100A	ssi120-4	TE1T200I	A06CICS1
br-rds-3	DS028005	ssi120-4	TE1T200I	A06CICS1
br-rds-3	DS032005	ssi120-4	TE1T200I	A06CICS1
br-rds-3	DS032010	ssi120-4	TE1T201I	A06CICS1
br-rds-3	DS032011	ssi120-4	TE1T202I	A06CICS1
br-rds-3	DS033005	ssi120-4	TE1T200I	A06CICS1
bracus	CICS1I	ssi120-4	TE1T202I	A06CICS1
bracus	DC0100A	ssi120-4	TE1T200I	A06CICS1
bracus	DS028005	ssi120-4	TE1T200I	A06CICS1
bracus	DS032005	ssi120-4	TE1T200I	A06CICS1
bracus	DS032010	ssi120-4	TE1T201I	A06CICS1
bracus	DS032011	ssi120-4	TE1T202I	A06CICS1
bracus	DS033005	ssi120-4	TE1T200I	A06CICS1
conappadapt	DC0102A	ssi120-6	TE1V002I	A06CICS1
mos-srv	CICS1I	ssi120-4	TE1T202I	A06CICS1
mos-srv	DC0100A	ssi120-4	TE1T200I	A06CICS1
mos-srv	DS028005	ssi120-4	TE1T200I	A06CICS1
mos-srv	DS032005	ssi120-4	TE1T200I	A06CICS1
mos-srv	DS032010	ssi120-4	TE1T201I	A06CICS1
mos-srv	DS032011	ssi120-4	TE1T202I	A06CICS1
mos-srv	DS033005	ssi120-4	TE1T200I	A06CICS1
osl-3030	DI0100A	ssi120-5	TE1T000I	A01ID01A
satkof1	IDMS15II	satkof1	TE1Z400I	A06ID15A
satkof1	IDMS7III	satkof1	TE1Z400I	A06ID07A
ultra04	DC0102A	ssi120-6	TE1V002I	A06CICS1

Figur 5: *WWW-basert oversikt over feilede forbindelser.*

Oversikten viser

- klient, symbolsk destinasjon, gateway og transaksjonshåndterer for feilede symbolske forbindelser.
- hvilke klienter som pinges.
- hvilke klienter hvis testapplikasjon er inkludert i analysen.
- hvilke klienter som er registrert av ekspertsystemet men som ikke har en gyldig test-applikasjon.

Denne informasjonen er viktig for at resultat fra ekspertsystemet skal tolkes korrekt. Ekspertsystemet rapporterer via WWW-grensesnittet illustrert i *figur 6*.



Analyse av forbindelser.

MULIGE FEILPUNKT.

Feilpunkt	Feil siden	Sannsynlighet	OK?
TE1Z400I–A06ID07A	16:14:31 09/12/1998	0.46	☐
TE1Z400I–A06ID15A	16:14:31 09/12/1998	0.44	☐
IDMS15II/satkof1	16:14:31 09/12/1998	0.44	☐
IDMS7III/satkof1	16:14:31 09/12/1998	0.31	☐
A06CICS1	15:24:26 09/12/1998	1.00	☐
DI0100A/ssi120–5	07:44:27 09/12/1998	0.44	☐
TE1T000I–A01ID01A	07:44:27 09/12/1998	0.44	☐

MULIGE FEILSCENARIER.

Feilscenario	Sannsynlighet
A06CICS1 IDMS15II/satkof1 IDMS7III/satkof1 TE1T000I–A01ID01A	0.08
A06CICS1 DI0100A/ssi120–5 IDMS15II/satkof1 TE1Z400I–A06ID07A	0.08
A06CICS1 TE1T000I–A01ID01A TE1Z400I–A06ID07A TE1Z400I–A06ID15A	0.08
A06CICS1 IDMS15II/satkof1 TE1T000I–A01ID01A TE1Z400I–A06ID07A	0.08
A06CICS1 DI0100A/ssi120–5 IDMS7III/satkof1 TE1Z400I–A06ID15A	0.08
A06CICS1 DI0100A/ssi120–5 TE1Z400I–A06ID07A TE1Z400I–A06ID15A	0.08
A06CICS1 IDMS7III/satkof1 TE1T000I–A01ID01A TE1Z400I–A06ID15A	0.08
A06CICS1 DI0100A/ssi120–5 IDMS15II/satkof1 IDMS7III/satkof1	0.08
A06CICS1 TE1Z400I–A06ID07A TE1Z400I–A06ID15A	0.04
A06CICS1 DI0100A/ssi120–5 IDMS15II/satkof1	0.04
A06CICS1 IDMS15II/satkof1 TE1Z400I–A06ID07A	0.04
A06CICS1 DI0100A/ssi120–5 TE1Z400I–A06ID15A	0.04
A06CICS1 TE1T000I–A01ID01A TE1Z400I–A06ID15A	0.04
A06CICS1 TE1T000I–A01ID01A TE1Z400I–A06ID07A	0.04
A06CICS1 IDMS15II/satkof1 TE1T000I–A01ID01A	0.04
A06CICS1 DI0100A/ssi120–5 TE1Z400I–A06ID07A	0.04
A06CICS1 IDMS15II/satkof1	0.02
A06CICS1 TE1T000I–A01ID01A	0.02
A06CICS1 DI0100A/ssi120–5	0.02
A06CICS1 TE1Z400I–A06ID15A	0.02

Oppdater analyse

Figur 6: *WWW-basert oversikt over feillokaliseringsresultat.*

Leseren kan merke seg følgende.

- ❑ Ekspertsystemet er basert et automatisk generert Bayesnettverk slik vi foreslo i *del 4*.
- ❑ Bayesnettverket genereres automatisk fra en topologidatabase hvert døgn. Dermed trenger ikke kunnskapsbasen vedlikeholdes manuelt ved strukturelle endringer i kommunikasjonssystemet.
- ❑ Alle komponenter hvis sannsynlighet for feil er større enn *0.1* gitt resultatet av pingene og tilstanden til de symbolske forbindelsene rapporteres.
- ❑ Grensesnittet mot ekspertsystemet er interaktivt ved at brukeren kan falsifisere foreslåtte alternativ. Dermed kan ekspertsystemet stegvis forfine feillokaliseringsresultatene i dialog med brukeren.

- ❑ Ekspertsystemet aktiveres automatisk hvert *10.* minutt slik at feiltidspunkt kan detekteres og registreres. Dette er viktig for å avgjøre feilenes alvorlighet.
- ❑ Foreslåtte feillokasjoner er lenket til topologidatabasen slik at brukeren lett kan slå opp relevant komponentinformasjon.
- ❑ De mest sannsynlige tilstandskonfigurasjonene fra den kombinerte sannsynlighetsfordelingen listes som feilsituasjoner (kun komponenter med feil listes).

Dette feillokaliserings- og omfangsbestemmelsesystemet er testet i fire måneder. Erfaringene som er gjort under testperioden beskrives under neste punkt.

6.2 Resultat av en fire måneder lang testperiode.

Vi har testet ut feillokaliseringssystemet i fire måneder. I testperioden benyttet to systemeksperter, de ansvarlige for gatewayene og kommunikasjonen over symbolske forbindelser, feillokaliseringssystemet daglig. Dette for å utelukke eller lokalisere feil. Ved hver feilsituasjon ble feilenes lokasjoner og resultat fra feillokaliseringssystemet registrert.

Testperioden ga i hovedtrekk følgende resultat.

- ❑ *51* av *57* feil ble lokalisert korrekt med det modifiserte deterministiske søket.
- ❑ De *6* oversette feilene ble oversett på grunn av støy; det modifiserte deterministiske søket kom aldri frem til noen tilstandskonfigurasjon. Systemekspertene så på dette som et relativt alvorlig problem.
- ❑ Det deterministiske søket ga alltid et resultat i løpet av *10* sekunder. Dermed var feillokaliseringen effektiv. Ved en feilsituasjon feilet en av hovedringene i SNA-nettverket (ring *N120*). Systemekspertene estimerte normal lokaliseringstid til *0.5 - 1* time. Feillokaliseringssystemet lokaliserte feilen i løpet av noen sekunder.
- ❑ Feillokaliseringssystemet håndterte heterogeniteten i underliggende kommunikasjonssystem; feil i SNA-nettverket kunne lokaliseres fra klientene i TCP/IP-nettverket.
- ❑ Feillokaliseringssystemet lokaliserte flere samtidige feil korrekt. I en av de korrekt håndterte feilsituasjonene feilet for eksempel *3* gatewayer samtidig.

Systemekspertene var fornøyd med feillokaliseringssystemet først og fremst fordi det var tidsbesparende, men også fordi det senker kompetansenivået nødvendig for effektiv feillokalisering.

6.3 Konklusjon.

Feillokaliserings- og omfangsbestemmelsesmetoden basert på det modifiserte deterministiske søket ser ut til å være av praktisk verdi. Hovedproblemet var at det modifiserte deterministiske søket ikke håndterer støy. Eksperimentene fra *del 5* viser derimot at det er god grunn til å tro at den nylig implementerte modifiserte Gibbs-sampleren vil håndtere støy bedre enn det modifiserte deterministiske søket.

7 Oppsummering og videre arbeid.

I denne delen oppsummerer vi først arbeidet som er gjort. Deretter skisserer vi videre arbeid.

7.1 Oppsummering.

I dette kapittelet foreslo vi et Bayesnettverk som modellerer årsakssammenhenger mellom tilstanden til tjenester, aktiveringssett, komponentsamarbeid og komponenter globalt i et kommunikasjonssystem. Beregninger i Bayesnettverk håndterer modellerte årsaks-

sammenhenger på et globalt plan. På denne måten har vi oppnådd global feillokalisering- og omfangsbestemmelse (**H1**).

Videre definerte vi et språk for spesifikasjon av sammensetningen til tjenester. Spesifikasjoner i dette språket kan genereres automatisk fra kommunikasjonssystemet eller fra konfigurasjons- og topologidatabaser. Et oversettelsesskjema ble benyttet til å generere foreslåtte Bayesnettverk automatisk fra gitte spesifikasjoner. Forsøk i eksempelkommunikasjonssystemet viste at oversettelsen er svært rask, så rask at Bayesnettverket kan genereres fra bunn av ved endringer i den strukturelle sammensetningen til et kommunikasjonssystem. Vi konkluderte dermed med at feillokaliserings- og omfangsbestemmelsesmetoden er adaptiv med hensyn på den strukturelle sammensetningen til et kommunikasjonssystem (**H2**).

Deretter foreslo vi to modifiserte beregningsmetoder for Bayesnettverk og sammenlignet disse med tradisjonelle beregningsmetoder. Nøyaktigheten til den best egnede beregningsmetoden, den modifiserte Gibbs-sampleren, ble testet under stigende grad av støy og et økende antall samtidige feil. Eksperimentene viste at den modifiserte Gibbs-sampleren håndterer støy, manglende måleresultat og ikke-totale feil tilfredstillende innenfor en tidsramme av *20* sekunder per feilsituasjon. Vi konkluderte dermed med at det er grunn til å tro at resulterende feillokaliserings- og omfangsbestemmelsesmetode kan lokalisere og bestemme omfanget til feil nøyaktig og effektivt (**H3**).

Til slutt undersøkte vi feillokaliserings- og omfangsbestemmelsesmetodens praktiske verdi. Vi implementerte et WWW-basert analysesystem for feillokalisering i Telenors TCP/IP-SNA gatewaysystemer. Under en testperiode på 4 måneder viste det seg at systemet var tidsbesparende, selv ved det modifiserte deterministiske søket. Ved innføring av den modifiserte Gibbs-sampleren som i følge eksperimentene håndterer støy og ikke-totale feil bedre enn det modifiserte søket, regner vi med at den praktiske verdien vil øke ytterligere. Vi konkluderte dermed med at det er god grunn til å tro at feillokaliserings- og omfangsbestemmelsesmetoden er av praktisk verdi.

Det gjenstår en rekke problemer vi ønsker å jobbe videre med. Disse problemene skisseres under neste punkt.

7.2 Videre arbeid.

Vi skisserer først mulige forbedringer ved spesifikasjonsspråket. Deretter gis en mer detaljert beskrivelse av hvordan dynamisk bestemte aktiveringssett kan håndteres. Videre skisseres innføring av komponentorienterte målinger i Bayesnettverket. Til slutt drøftes mulige anvendelser for den modifiserte Gibbs-sampleren.

Utvidelser i spesifikasjonsspråket.

Spesifikasjonsspråket foreslått i *del 3* har vist seg å være noe stivbent. Man kan for eksempel ikke navngi tjenester. Videre støtter språket kun symmetriske samarbeidstyper. Vi ønsker å undersøke hvordan uttrykkskraften til språket kan forbedres slik at det håndterer en større klasse tjenester.

Håndtering av dynamisk bestemte aktiveringssett.

Videre støtter oversettelsesskjemaet definert i *tabell 1* kun statisk bestemte aktiveringssett.

Ved alternative aktiveringssett må vi innføre *eller-kjeder* i Bayesnettverket tilsvarende og-kjedene. Et slikt oversettelsesskjema defineres i *tabell 2*.

Produksjon	Semantiske regler
Kommunikasjonssystem -> Tjenesteliste	
Tjenesteliste -> Tjeneste **;** *Tjenesteliste*	
Tjenesteliste -> Tjeneste **;**	
K -> ***Komp*** **:** ***Type***	$\mathbf{P}_{\boldsymbol{Type}}$**(*Komp*)**; *K* := ***Komp***;

K -> ***Komp*** **:** ***Type*** **(** *Tjeneste* **)** ***Komp'***	$\mathbf{P}_{\boldsymbol{Type}}$**(*Komp*)**;
	$\mathbf{P}_{\boldsymbol{Type}}$**(*Komp'*)**;
	Hvis !forrige(***Komp*()*Komp'***) så {
	forrige(***Komp*()*Komp'***) := *Tjeneste*;
	behandlet(***Komp*(***Tjeneste***)*Komp'***);
	} ellers hvis !behandlet(***Komp*(***Tjeneste***)*Komp'***) så {
	$\mathbf{P}_{\mathbf{eller}}$**(*Komp*(***Tjeneste***)*Komp'* |** forrige(***Komp*()*Komp'***),** *Tjeneste***);**
	forrige(***Komp*()*Komp'***) := ***Komp***(*Tjeneste)****Komp'***;
	}
	$\mathbf{P}_{\boldsymbol{Type}\mathbf{rel}}$**(*Komp*()*Komp'*
		Komp,** forrige(***Komp*()*Komp'***),** ***Komp'***);
		K := ***Komp*** **()** ***Komp'***;

Tjeneste -> K **~** *Tjeneste'*	$\mathbf{P}_{\mathbf{og}}$**(***K*~*Tjeneste'* **|** *Tjeneste'***,** *K***)**; *Tjeneste* := *K* **~** *Tjeneste'*;
Tjeneste -> K	*Tjeneste* := *K*;

Tabell 2: *Et oversettelsesskjema for oversetting av strenger i spesifikasjonsspråket til et Bayesnettverk for feillokalisering og omfangsbestemmelse.*

Oversettelsskjemaet skal tolkes som beskrevet i *del 3*, men med følgende modifikasjoner. Funksjonen ‘forrige()’ angir forrige spesifiserte alternativ i gitt eller-kjede. Funksjonen ‘behandlet()’ returnerer sann hvis det er første gang den kalles med gitt parameter, ellers returnerer den falsk. Funksjonen ‘!’ inverterer sannhetsverdien til parameteret.

Man får gjerne dynamisk bestemte aktiveringssett i feiltolerante adaptive systemer. Et av målene til et adaptivt og feiltolerant kommunikasjonssystem er å hindre at tjenestene påvirkes av feil. Dermed vil det være vanskelig å lokalisere feil kun fra tilstanden til tjenester.

Vi antar at aktiveringshyppigheten til et aktiveringssett vil falle ved feil innenfor aktiveringssettet. Vi foreslår derfor at aktiveringshyppigheten beregnes adaptivt av feillokaliserings- og omfangsbestemmelsesmetoden. Da kan aktiveringssett hvis aktiveringsrate faller feilmeldes i Bayesnettverket. På denne måten kan tilstanden i et kommunikasjonssystem med adaptivt bestemte aktiveringssett beregnes.

Under videre arbeid ønsker vi å undersøke om dette er tilfellet.

Innføring av komponentorienterte målinger i Bayesnettverket.

Feillokaliserings- og omfangsbestemmelsesmetoden er som nevnt tjenesteorientert. Ved å innføre komponentorienterte målinger i Bayesnettverket kan det være mulig å oppnå et mer detaljert og mer nøyaktig estimat av tilstanden i et kommunikasjonssystem.

Etter som årsakssammenhengene mellom resultatet av komponentmålinger i et kommunikasjonssystem er komplekse og lite forstått [10] antar vi at en innlemmelse av komponentorienterte målinger i Bayesnettverket vil være et omfattende og spennende arbeide.

Videre undersøkelser og arbeid med den modifiserte Gibbs-sampleren.

Til slutt vil vi undersøke hvordan den modifiserte Gibbs-sampleren håndterer generelle Bayesnettverk. Universitetet i Ålborg, Danmark, har samlet et sett av store og komplekse Bayesnettverk til dette formålet. Det er interessant å se hvilke følger antagelsen om en "tre-struktur" innenfor blokkene vil få når antagelsen ikke holder. Videre ønsker vi å undersøke egenskapene til beregningsmetoden når den nøyaktig beregningsmetoden innenfor blokker for eksempel byttes ut med et stokastisk søk i årsaksretningen.

8 Referanser.

[1] S. K. Goyal, AI in Support of Distributed Network Managment, Network and Distributed Systems Management, Addison Wesley, 1994, s. 545-546.
[2] Uyless Black, Network Management Standards: The OSI, SNMP and CMOL Protocols, McGraw-Hill, Inc, 1992.
[3] Judea Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Publishers Inc., 1991.
[4] Stuart J. Russell og Peter Norvig, Artificial intelligence : a modern approach, Prentice Hall, 1995.
[5] Finn V. Jensen, An Introduction to Bayesian Networks, 1996.
[6] Enrique Castillo, Expert systems and probabilistic network models, Springer, 1997.
[7] B.D. Ripley, Pattern recognition and neural networks, Cambridge University Press, 1996.
[8] A. V. Aho, R. Sethi og J.D. Ullman, Compilers: Principles, Techniques and Tools, Addison-Wesley Publishing Company, 1986.
[9] Vladimir Cherkassky, Fuzzy Inference Systems: A Critical Review, Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, 1998, s. 177-197.
[10] Cynthia S. Hood og Chuanyi Ji, Proactive Network Fault Detection, IEEE INFOCOM’97, s. 1147-1155.

Kapittel 3: Et forslag til en metode for automatisk, lokal og adaptiv feildeteksjon og feilkorrigering i kommunikasjonssystemer.

Ole-Christoffer Granmo,

Institutt for Informatikk,

Universitetet i Oslo.

10/2/99

1 Innledning.

I dette kapittelet foreslår vi en metode for automatisk, lokal og adaptiv feildeteksjon og feilkorrigering ved overvåkning av kommunikasjonssystemer.

Et kommunikasjonssystem overvåkes ved å måle *tjenesteattributter* og *komponentattributter*. En tjenesteattributt beskriver egenskaper ved en tjeneste, for eksempel tjenestens tilgjengelighet eller utførelsestid. En komponentattributt beskriver egenskaper ved en komponent, for eksempel belastningen på et transmisjonsmedium eller antall detekterte rammefeil i nettlaget til en ruter.

Poenget med *målingene* er at de skal gi et *dekkende bilde* av tilstanden i kommunikasjonssystemet. Med en måling mener vi her selve målefunksjonen. Med et dekkende bilde menes at alle de kritiske egenskapene i kommunikasjonssystemet måles så hyppig at ikke resultatene er foreldet ved bruk. Hvis bildet er dekkende vil feil kunne detekteres via endringer i måleresultatene. Feildeteksjon består dermed i å analysere *måleresultat* for så å avgjøre om de indikerer feil. Med måleresultat mener vi resultatet av en måling.

Når en feil er lokalisert enten direkte ved deteksjon eller ved hjelp av en lokaliseringsmetode, må det avgjøres hvordan feilen skal korrigeres. Feilkorrigeringshandlinger kan grovt sett klassifiseres som følger etter stigende kostnad: omkonfigurering og omstart av komponenter, oppgradering av komponenter og til slutt fysisk omkonfigurering eller utvidelser i kommunikasjonssystemet. De fleste feil kan korrigeres ved omkonfigurering og omstart av komponentene.

Metoden som foreslås i dette kapittelet består av et rammeverk for *lokal feildeteksjon og feilkorrigering* samt en algoritme for automatisk og adaptiv feildeteksjon og feilkorrigering. Med lokal feildeteksjon og feilkorrigering menes her feildeteksjon og feilkorrigering i enkeltkomponenter og enkelttjenester og ikke sammensetninger av disse.

Vi tenker oss at rammeverket skal danne et uniformt grunnlag for feildeteksjon og feilkorrigering. Det uniforme grunnlaget oppnås for det første ved et språk for spesifikasjon av feildeteksjons- og feilkorrigeringsgrensesnitt mot komponenter og tjenester med tilhørende målinger, feildeteksjonshandlinger og feilkorrigeringshandlinger. For det andre består rammeverket av en feildeteksjons- og feilkorrigeringsdatabase automatisk generert fra en spesifikasjon i spesifikasjonsspråket. Feildeteksjons- og feilkorrigeringsdatabasen vil inneholde feildeteksjons- og feilkorrigeringsdata på et uniformt format.

Videre tenker vi oss at feildeteksjons- og feilkorrigeringsalgoritmen skal bruke feildeteksjons- og feilkorrigeringsdataene i feildeteksjons- og feilkorrigeringsdatabasen til å utføre målinger, feildeteksjonshandlinger og feilkorrigeringshandlinger i et kommunikasjonssystem. Feildeteksjons- og feilkorrigeringsalgoritmen velger da feilde-

teksjons- og feilkorrigeringshandlinger ved å se på lagrede *handlingseksempler* i feildeteksjons- og feilkorrigeringsdatabasen. Med et handlingseksempel menes her et måleresultat fra en gitt situasjon knyttet til en egnet *handling* for situasjonen. Med en handling menes en feildeteksjonshandling eller en feilkorrigeringshandling. Vi oppnår dette ved å basere algoritmen på en klasse statistiske mønstergjenkjenningsmetoder kalt nærmeste-nabo metoder.

Før vi går i gang med utledningen av rammeverket og algoritmen drøftes de egenskapene som vi under arbeidet med Telenors TCP/IP-SNA gatewaysystemer (se *kapittel 2*) har erfart som viktige ved automatisk feildeteksjon og feilkorrigering. Deretter gis en oversikt over kapittelet.

Viktige egenskaper.

1. Når en automatisk feildeteksjons- og feilkorrigeringsmetode først utfører en handling automatisk bør det være relativt sikkert at denne handlingen er egnet. Det er for eksempel ikke særlig heldig hvis en automatisk feilkorrigeringsmetode utfører en vilkårlig handling, for eksempel omstart av en komponent, ved *ukjente måleresultat*. Med et ukjent måleresultat menes her et måleresultat hvor beslutningsgrunnlaget for håndtering av måleresultat ikke omfatter måleresultatet. Videre er det ikke alltid ønskelig at en handling skal utføres automatisk hvis det eksisterer tvil mellom to eller flere handlinger.
2. En annen egenskap vi har erfart som viktig er at en automatisk feildeteksjons- og feilkorrigeringsmetode bør håndtere både spesifikk kunnskap om enkeltkomponenter og enkelttjenester samt generell kunnskap om komponenttyper og tjenestetyper. Ofte skal komponenter eller tjenester av samme type håndteres likt, men lokale forskjeller kan også være avgjørende. Dette er spesielt tilfellet når *miljøet* til en komponent eller tjeneste virker inn på hvilke handlinger som er best egnet. Med miljøet til en komponent eller tjeneste menes her resten av kommunikasjonssystemet. I Telenors TCP/IP-SNA gatewaysystemer måles for eksempel tilgjengeligheten og responstiden for overføring av data fra klienter i TCP/IP-nettverket til tjenere i SNA-nettverket. Responstiden er her avhengig av miljøet rundt tjenesten og ikke bare tjenestens type (dvs hva som overføres og hvordan dette gjøres).
3. Til slutt ser vi at en automatisk feildeteksjons- og feilkorrigeringsmetode bør være adaptiv. I overvåkningssystemet HP OpenView [11], som benyttes til å overvåke Telenors TCP/IP-nettverk, knyttes statiske hvis-så regler og terskelverdier til automatisk utførte målinger. På denne måten oppnås automatisk feildeteksjon og eventuelt også feilkorrigering. Det er to ulemper med denne typen statiske fremgangsmåter. Hvis en feiltype overses eller spesifiseres galt får dette direkte og varige konsekvenser. I beste fall oppdages mangelen på et tidlig tidspunkt. I verste fall forblir feildeteksjonen og feilkorrigeringen suboptimal. Dette er en alvorlig ulempe fordi det er vanskelig å forutse hvordan feil kan detekteres og korrigeres i komplekse kommunikasjonssystemer. Den andre ulempen er at mange feildeteksjons- og feilkorrigeringsmiljøer endrer seg dynamisk. Det betyr for eksempel at hvis-så regler og terskelverdier som passer ved et tidspunkt ikke nødvendigvis passer ved et senere tidspunkt. En terskel for unormal tjenesteresponstid kan for eksempel bli utdatert når bruksmønsteret eller topologien i kommunikasjonssystemet endres. Det er dermed ønskelig at en automatisk feildeteksjons- og feilkorrigeringsmetode tilpasser seg et gitt miljø adaptivt.

Oversikt over resten av kapittelet.

I *del 2* definerer vi et språk for spesifikasjon av feildeteksjons- og feilkorrigeringsmiljøer. Deretter i *del 3* beskrives hvordan en streng i spesifikasjonsspråket automatisk kan over-

settes til en feildeteksjons- og feilkorrigeringsdatabase. Dermed er rammeverket for feildeteksjon og feilkorrigering definert. I *del 4* foreslår vi en automatisk og adaptiv feildeteksjons- og feilkorrigeringsalgoritme som interakterer med et kommunikasjonssystem via definert rammeverk. Frem til *del 5* vil vi presentere et noe forenklet bilde av feildeteksjon og feilkorrigering. Dette for å gi et klart bilde av valgene som gjøres. I *del 5* tar vi derimot for oss noen av de underliggende utfordringene. Til slutt i *del 6* konkluderer vi med en oppsummering av arbeidet.

2 Et språk for spesifikasjon av feildeteksjons- og feilkorrigeringsmiljøer.

For å kunne utføre hensiktsmessig feildeteksjon og feilkorrigering må man kjenne hvilke komponenter, tjenester, målinger og handlinger som er tilgjengelig i kommunikasjonssystemet samt betydningen av disse. I heterogene kommunikasjonssystemer er dette vanskelig fordi både den syntaktiske og semantiske delen av grensesnittet mot kommunikasjonssystemet kan variere. I Telenors TCP/IP-SNA gatewaysystemer må man for eksempel ved feildeteksjon og feilkorrigering kjenne to vidt forskjellige miljøer. Dette begrenser mulighetene for automatisk feildeteksjon og feilkorrigering. Vi foreslår derfor her et språk for spesifikasjon av feildeteksjons- og feilkorrigeringsmiljøer. Det er meningen at dette språket skal brukes til å gi et forenklet bilde av feildeteksjons- og feilkorrigeringsmiljøer slik at automatisk og adaptiv feildeteksjon og feilkorrigering understøttes.

2.1 Metodevalg.

Her drøftes de forenklinger vi har gjort.

I et kommunikasjonssystem vil komponenter og tjenester være av en gitt type. Ved å knytte målinger og handlinger mot typene i stedet for komponentene og tjenestene begrenses “tolkningsrommet”. Dette fremmer automatisk læring og samordning av handlingseksempler.

Målinger og handlinger kan spesifiseres ved *atomære funksjoner* eller ved *ikke-atomære funksjoner*. Med atomære funksjoner mener vi her funksjoner med *en* innvariabel og *en* utvariabel. Målefunksjoner har måleresultat som utverdi. Handlingfunksjoner har handlingsresultat som utverdi. Begge har komponentnavn eller tjenestenavn som innverdi. Ved å begrense funksjonsrommet til atomære funksjoner forenkles spesifikasjonsspråket; en måling eller handling kan spesifiseres kun ved å navngi tilhørende funksjon. Dette åpner for uniform tilknytning av måleresultat til handlinger (se *del 4*). Ulempen er at ikke-atomære funksjoner må omformes til atomære funksjoner ved spesifikasjon.

Under neste punkt defineres spesifikasjonsspråket.

2.2 Spesifikasjonsspråket.

Spesifikasjonsspråket defineres som en kontekstfri grammatikk. Hvordan vi bruker kontekstfrie grammatikker ble beskrevet i *kapittel 2*.

Vi lar et feildeteksjons- og feilkorrigeringsmiljø bestå av komponenter og tjenester. Hver komponent og tjeneste har en gitt type. Denne typen avgjør hvilke målinger og handlinger som kan utføres mot komponenten eller tjenesten. I språket deklareres typene først. Deretter deklareres komponentene og tjenestene:

Miljø -> TypeDeklListe **;** *DeklListe*

TypeDeklListe -> TypeDekl **,** *TypeDeklListe*
TypeDeklListe -> TypeDekl

DeklListe -> Dekl ***,*** *DeklListe*
DeklListe -> Dekl.

Hver type gis et navn, ***TypeNavn***. Dette navnet knytter vi til målingene, *MålingListe*, og handlingene, *HandlingListe*, som kan utføres mot komponenter eller tjenester av denne typen. I tillegg innfører vi tre attributter, ***TvilVerdi***, ***UkjentVerdi*** og ***MinneVerdi***. Attributtenes betydning kommer klart frem i *del 4*.

TypeDekl ->
TypeNavn (*MålingListe* ***;*** *HandlingListe* ***; TvilVerdi, UkjentVerdi, MinneVerdi)***;

En *MålingListe* er en liste med navn, ***MålingNavn***, som navngir atomære målefunksjoner:

MålingListe -> ***MålingNavn ,*** *MålingListe*
MålingListe -> ***MålingNavn***.

En *HandlingListe* er en liste med navn, ***HandlingNavn***, som navngir atomære handlingfunksjoner:

HandlingListe -> ***HandlingNavn ,*** *HandlingListe*
HandlingListe-> ***HandlingNavn***.

Deklarasjon av en komponent eller tjeneste består av navnet, ***Navn***, samt typen, ***Type***:

Dekl -> ***Navn : Type***.

Her følger to eksempler på strenger i spesifikasjonsspråket. Leseren trenger her ikke forstå spesifiserte feildeteksjonsmiljøer i sin fulle dybde.

I Telenors løsning for overvåkning av brannmurer måles ulike egenskaper ved trafikken fra hver kildeadresse. Disse målingene danner grunnlaget for rapporthandlinger. Spesifikasjonen er som følger:

BrannmurKilde(
AntallTidsenheterMellomAksepterteAksesser, AntallTidsenheterMellomStoppedeAksesser, VariasjonIAntallTidsenheterMellomAksesser, AndelForskjelligeTjenester, AndelForskjelligeDestinasjoner;
Ignorer, RapporterFeilKonfigurasjon, RapporterUnormalTjenestebruk, RapporterTjenesteScan, RapporterDestinasjonsScan;
0.7, 0.01, 100
) ;

134.47.111.12 : BrannmurKilde,
134.47.200.07 : BrannmurKilde,
...,
147.100.104.12 : BrannmurKilde

Det andre eksempelet er tatt fra Telenors TCP/IP-SNA gateway-løsninger (se *kapittel 2*). Her måles tilgjengeligheten og responstiden på ende-til-ende forbindelser. Disse må-

lingene kan brukes til å avgjøre om en forbindelse fungerer eller om det eksisterer et problem. Spesifikasjonen er som følger:

EndeTilEndeForbindelse(
Tilgjengelighet, Responstid;
RapporterOK, RapporterMuligProblem;
0.5, 0.0, 100
);

CICS17I/al-14()CICS17I/ssi120~CICS17I/ssi120()A01CIC17 : EndeTilEndeForbindelse,
...,
DS032012/al-14()DS032012/ssi120~DS032012/ssi120()A01CIC18 : EndeTilEndeForbindelse

Merk at dette feildeteksjonsmiljøet kan knyttes til feillokaliseringssystemet beskrevet i *kapittel 2*, *del 6*.

2.3 Oppsummering.

Vi har nå definert et språk for spesifikasjon av feildeteksjons- og feilkorrigeringsmiljøer. I neste del foreslår vi hvordan en feildeteksjons- og feilkorrigeringsdatabase kan genereres automatisk fra en streng i dette språket.

3 Generering av en feildeteksjons- og feilkorrigeringsdatabase.

I denne delen foreslår vi hvordan en distribuert feildeteksjons- og feilkorrigeringsdatabase kan genereres automatisk fra en streng i det definerte spesifikasjonsspråket. Vi tenker oss at denne databasen sammen med spesifikasjonsspråket skal danne et rammeverk for automatisk feildeteksjon og feilkorrigering.

Databasen skal inneholde feildeteksjons- og feilkorrigeringsinformasjon på et uniformt format. Mer presist skal databasen inneholde typen til komponenter og tjenester, tilgjengelige målinger og handlinger for hver type, ulike feildeteksjons- og feilkorrigeringsregler og til slutt lagrede handlingseksempler knyttet til hver komponent og tjeneste.

Her defineres kun innholdet av databasen. Vi tar for oss distribusjonen med hensyn på typer, komponenter og tjenester i *kapittel 4*.

3.1 Et oversettelsesskjema for generering av en feildeteksjons- og feilkorrigeringsdatabase.

Feildeteksjons- og feilkorrigeringsdatabase genereres ved hjelp av et oversettelsesskjema (se *kapittel 2* for en kort beskrivelse av hvordan vi bruker oversettelsesskjemaer):

Produksjon	Semantiske regler
Miljø -> TypeDeklListe **;** *DeklListe*	
TypeDeklListe -> TypeDekl **,** *TypeDeklListe*	
TypeDeklListe -> TypeDekl	
DeklListe -> Dekl **,** *DeklListe*	

DeklListe -> Dekl	
TypeDekl -> ***TypeNavn*** **(** *MålingListe* **;** *HandlingListe* **;** ***TvilVerdi*** **;** ***UkjentVerdi*** **;** ***MinneVerdi*** **)**	$MListe(TypeNavn) \leftarrow MålingListe;$ $HListe(TypeNavn) \leftarrow HandlingListe;$ $Tvil(TypeNavn) \leftarrow TvilVerdi;$ $Ukjent(TypeNavn) \leftarrow UkjentVerdi;$ $Minne(TypeNavn) \leftarrow MinneVerdi;$
MålingListe -> ***MålingNavn*** **,** *MålingListe'*	*MålingListe* := ***MålingNavn***, *MålingListe'*;
MålingListe -> ***MålingNavn***	*MålingListe* := ***MålingNavn***;
HandlingListe -> ***HandlingNavn*** **,** *HandlingListe'*	*HandlingListe* := ***HandlingNavn***, *HandlingListe';*
HandlingListe -> ***HandlingNavn***	*HandlingListe* := ***HandlingNavn***;
Dekl -> ***Navn*** **:** ***TypeNavn***	$Type(Navn) \leftarrow TypeNavn;$ $ForAlle$ $h \in HListe(TypeNavn)\ (Res(Navn, h)$ $\leftarrow RingBuffer(Minne(TypeNavn)));$ $NavneListe(TypeNavn) \leftarrow$ $NavneListe(TypeNavn), Navn;$

I oversettelseskjemaet deklareres funksjonsverdien til funksjon *f* for argumentsymbol *s* til å være lik *t* med $f(s) \leftarrow t$.

Oversettelseskjemaet deklarerer "skjelettfunksjonene" *MListe*, *HListe*, *Tvil*, *Ukjent*, *Minne*, *Type*, *NavneListe* og *Res*. Funksjonene benyttes av feildeteksjons- og feilkorrigeringsalgoritmen som foreslås i *del 4*. Her følger en kort beskrivelse.

1. *MListe* knytter hvert typenavn til en liste med navnet på atomære målefunksjoner.
2. *HListe* knytter hvert typenavn til en liste med navnet på atomære handlingfunksjoner.
3. *Tvil* knytter hvert typenavn til en *TvilVerdi* hvis betydning beskrives i *del 4*.
4. *Ukjent* knytter hvert typenavn til en *UkjentVerdi* som beskrives i *del 4*.
5. *Minne* knytter hvert typenavn til en *MinneVerdi*. *Minneverdien* bestemmer antall handlingseksempler som skal kobles til hver handling mot en komponent eller tjeneste.
6. *Type* knytter hvert komponentnavn og tjenestenavn til en type.
7. *NavneListe* knytter hver typenavn til et liste med navnet til alle komponenter eller tjenester av gitt type.
8. *Res* knytter navnet på en handlingfunksjon og navnet på en komponent eller tjeneste, til et ringbuffer med størrelse bestemt av funksjonen *Minne*. *RingBuffer* er en funksjon

som returnerer et *ringbuffer* med gitt størrelse. Med et ringbuffer menes et buffer som skyver ut de tidligst innsatte elementene ved innsetting av nye når bufferet er fullt.

3.2 Oppsummering.

Vi har nå beskrevet hvordan en feildeteksjons- og feilkorrigeringsdatabase kan genereres automatisk fra en streng i spesifikasjonsspråket fra *del 2*. Resulterende rammeverk skal danne et forenklet miljø for feildeteksjon og feilkorrigering med tanke på automatisering. I neste del foreslår vi en automatisk og adaptiv feildeteksjons- og feilkorrigeringsalgoritme som interakterer med et spesifisert miljø ved hjelp av dette rammeverket.

4 En algoritme for automatisk og adaptiv feildeteksjon og feilkorrigering.

I denne delen foreslår vi en algoritme for automatisk og adaptiv feildeteksjon og feilkorrigering.

En systemekspert eller systemadministrator detekterer gjerne feil ved å tolke måleresultat fra komponenter og tjenester. En tolkning kan være basert på kunnskap om måleresultatenes betydning, for eksempel hvilke komponent- eller tjenestetilstander som kan ha forårsaket gitte måleresultat. Det er også viktig å vurdere hvordan miljøet til en komponent eller tjeneste virker inn.

Videre bestemmer gjerne en systemekspert eller systemadministrator egnede handlinger ved å vurdere følgene av handlingene.

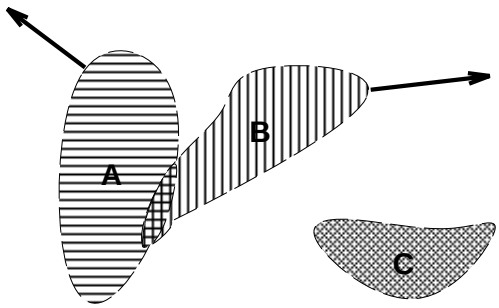
Beskrevet fremgangsmåte kan automatiseres ved å for eksempel konstruere en årsaksmodell som representerer en komponents interne struktur knyttet til måleresultat. Her har vi derimot mindre uniform kvalitativ kunnskap om årsaksammenhenger enn det vi har ved tjenesteorientert feillokalisering og omfangsbestemmelse. Vi foreslår derfor i stedet at *en direkte avbildning fra måleresultat til egnede handlinger* approksimeres fra handlingseksempler. Med en direkte avbildning fra måleresultat til egnede handlinger menes her avbildningen man får ved å koble måleresultat fra alle mulige feilsituasjoner til den optimale handlingen i hver situasjon. Den foreslåtte fremgangsmåten innebærer at handlingsvalg foretas ved ren mønstergjenkjenning.

Først, under *4.1*, ser vi på noen problemer som bør håndteres ved bruk av approksimative avbildningsmetoder til lokal feildeteksjon og feilkorrigering. Deretter under *4.2* drøftes fordelene og ulempene ved ulike approksimative avbildningsmetoder med hensyn på beskrevne problemer. Videre foreslår vi en approksimativ avbildningsmetode hvor modellparametrene muliggjør eksplisitt trimming med hensyn på beskrevne problemer. Drøftingen ender under *4.3* opp i et forslag til en algoritme for automatisk og adaptiv feildeteksjon og feilkorrigering. Algoritmen er knyttet til databasen foreslått i *del 3*. Til slutt under *4.4* oppsummeres denne delen.

4.1 En konkretisering av viktige egenskaper en metode for automatisk feildeteksjon og feilkorrigering bør ha.

Her konkretiseres egenskapene som ble drøftet i *del 1* med tanke på bruk av approksimative avbildningsmetoder til lokal feildeteksjon og feilkorrigering i kommunikasjonssystemer.

På grunn av kompleksiteten i kommunikasjonssystemer kan man sjelden regne en approksimative avbildning basert på handlingseksempler som komplett. Det betyr at en avbildning må kunne oppdateres stegvis etter hvert som nye handlingseksempler blir tilgjengelig. Videre betyr dette at en approksimativ avbildningsmetode må kunne detektere ukjente måleresultat. *Figur 1* illustrerer blant annet dette problemet.



Figur 1: *Den optimale avbildningen fra to-dimensjonale måleresultat til handlingene A, B og C.*

I figuren illustreres den optimale avbildningen fra to-dimensjonale måleresultat til handlingene *A*, *B* og *C*. Anta at avbildningen fra måleresultat til handlingene *A* og *B* allerede er approksimert fra handlingseksempler. Anta videre at avbildningen fra måleresultat til handling *C* initielt ikke er kjent. Når måleresultat som optimalt skal avbildes til *C* måles bør disse målresultatene registreres som ukjente og ikke avbildes til handling *A* eller *B*. Videre bør tvil mellom to eller flere handlinger kunne detekteres. I *figur 1* vil det for eksempel oppstå tvil mellom handling *A* og *B* ved måleresultat i området hvor avbildningsdomenet til *A* og *B* overlapper. Disse eksemplene er en konkretisering av egenskap *1* fra *del 1*.

I figuren illustreres også en konkretisering av egenskap *3* fra *del 1*. Man må for eksempel kunne inkorporerer handlingseksempler fra handling *C* i den eksisterende approksimative avbildningen etter hvert som de blir tilgjengelig. Dette for at avbildningsmetoden adaptivt skal kunne lære seg avbildningsdomenet til ulike handlinger. Videre kan som nevnt miljøet til en komponent eller tjeneste gjerne endre seg over tid. En avbildning fra måleresultat til normal eller unormal responstid kan for eksempel bli utdatert når bruksmønsteret eller topologien i kommunikasjonssystemet endres. *Figur 1* illustrerer dette ved at den optimale avbildningen mot *A* og *B* over tid beveger seg i pilenes retning. Hvis avbildningsmetoden ikke håndterer slike endringer adaptivt vil approksimert avbildning i stadig større grad avvike fra den optimale avbildningen.

Under neste punkt ser vi på hvordan ulike approksimative avbildningsmetoder håndterer disse problemene. Videre foreslår vi en avbildningsmetode hvis parametre kan trimmes eksplisitt med hensyn på beskrevne problemer.

4.2 Metodevalg.

For å finne en approksimativ avbildning fra måleresultat til egnede handlinger ved hjelp av handlingseksempler kan vi velge mellom metoder som Kohonens selvorganiserende kart eller Learning vector quantization (LVQ), feed-forward nevrale nettverk, radial-basis nettverk eller andre fleksible paramtriske modeller hvor det i liten grad gjøres antagelser om avbildningens form, parametriske modeller hvor det gjøres parameterbegrensende antagelser om avbildningens form, eller nærmeste-nabo metoder hvor avbildnigsverdier beregnes direkte fra handlingseksemplene (se [12] for en beskrivelse av disse metodene).

Fleksible parametriske avbildningsmodeller.

Fordelen ved Kohonens selvorganiserende kart/LVQ (strengt tatt klassifiseres ikke disse som parametriske men har mange av de samme egenskapene), feed-forward nevrale nettverk, radial-basis nettverk, og andre fleksible parametriske modeller er at det i liten grad

gjøres noen antagelser om formen på avbildningen som skal approksimeres. Videre beregnes approksimerte avbildningsverdier ofte raskt.

Ulempen ved fleksible parametriske avbildningsmodeller synes å være at man trenger et stort antall på forhånd tilgjengelig handlingseksempler. Videre er fastsettelsen av modellparametre svært krevende. I [8] beskrives for eksempel en metode for *korrelering av alarmer* ved hjelp av Kohonens selvorganiserende kart. Med korrelering av alarmer menes samordning av flere måleresultat slik at feil kan detekteres og eventuelt lokaliseres nøyaktig. Den beskrevne fremgangsmåten i [8] krevde for det første et stort sett av eksempler. Disse måtte generes ved å simulere feil. Dermed vil resulterende approksimative avbildning trolig kun gi gode resultat ved forutsette feilsituasjoner. Videre krevde fastsettelsen av modellparametre *100 000* justeringer før den gjennomsnittlige feilraten falt under *1%* for trenings/verifikasjons-eksemplene.

Vi konkluderer derfor med at det er god grunn til å tro at beskrevne svakheter ved fleksible parametriske modeller gjør dem lite adaptive med hensyn på endringer i den optimale underliggende avbildningen fra måleresultat til egnede handlinger; både en krevende eksempelgenerering og en krevende fastsettelse av modellparametre må utføres i takt med endringer.

Ved visse modifikasjoner kan derimot ukjente måleresultat detekteres.

Parametriske avbildningsmodeller hvor det gjøres parameterbegrensende antagelser om avbildningens form.

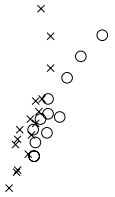
Ved å anta en gitt form på avbildningen kan man slippe unna med få modellparametre. Få modellparametre fører blant annet til en mindre krevende justeringsprosess. Problemet er at man i vårt tilfelle ikke kan gjøre noen antagelser om avbildningenes form. Vi ser derfor bort i fra denne typen enkle parametriske modeller.

Nærmeste-nabo metoder.

Ved nærmeste-nabo metoder approksimeres avbildningsverdien til et måleresultat direkte fra de *nærmeste* måleresultatene blant handlingseksemplene. Nærhet defineres etter et eller annet avstandsmål, f.eks. Euklidsk avstand. Fordelen ved nærmeste nabo-metoder er at det ikke gjøres noen antagelser om formen på avbildningen. Videre beregnes den approksimerte avbildningen direkte fra handlingseksemplene. Dermed får man ingen tidkrevende treningsfase med justering av modellparametre. Problemet er at alle handlingseksemplene må lagres samt at beregningen av avbildningsverdier kan være tidkrevende ved et stort antall lagrede handlingseksempler.

For å begrense antall handlingseksempler ved nærmeste nabo-metoder finnes det ulike strategier for editering av eksempelmaterialet (se [12]). En metode, *kondensering*, består i å kun lagre et handlingseksempel hvis de allerede lagrede eksemplene ikke avbilder måleresultatet i handlingseksempelet korrekt. Ved å iterere over ikke-lagrede handlingseksempler vil man til slutt få et subsett av lagrede handlingseksempler som også avbilder de ulagrede handlingseksemplene korrekt. En slik editering kan i vårt tilfelle best utføres stegvis, dvs etter hvert som handlingseksemplene blir tilgjengelig. Varianter av denne fremgangsmåten danner grunnlaget for de “case”-baserte resonneringssystemene for feilhåndtering beskrevet i artiklene [9] og [10].

Figur 2 illustrerer hvordan lagrede handlingseksempler fra avbildningen i *figur 1* kan fordele seg ved stegvis kondensering.



Figur 2: *Kondensering av handlingseksempler fra handling A og B.*

I figuren ser vi hvordan de lagrede handlingseksemplene grenser opp måleresultatene som skal avbildes til *A* eller *B* (se *figur 1*). Det er i vår sammenheng tilsynelatende tre problemer ved denne metoden.

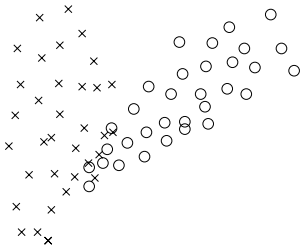
For det første vil strategien føre til en over tid ubegrenset konsentrasjon av lagrede handlingseksempler i området hvor avbildningsdomenet til *A* og *B* overlapper. Dette fordi en approksimert avbildning vil avbilde måleresultat i området med overlapping galt relativt hyppig. Dermed lagres de galt avbildede handlingseksemplene.

For det andre er det ikke mulig å identifisere ukjente måleresultat med hensyn på lagrede handlingseksempler da den opprinnelige fordelingen av måleresultat ikke er bevart. Hvis for eksempel måleresultat som skal avbildes til *C* måles, vil måleresultatet avbildes til *B* ved handlingseksemplene i figuren.

Til slutt er det vanskelig å avgjøre hvordan grensene skal forflytte seg ved gradvise endringer i den optimale avbildningen.

Forslag til nærmeste-nabo basert avbildningsmetode.

Vi foreslår derfor her at antall lagrede handlingseksempler begrenses til de *n* siste eksemplene fra hver handling. Resulterende lagring av handlingseksempler illustreres i *figur 3*.



Figur 3: *De n siste handlingseksemplene for handling A og handling B.*

Det synes å være to fordeler ved foreslått strategi.

For det første vil man kunne detektere ukjente måleresultat fra fordelingen til de kjente måleresultatene. Hvis måleresultat som skal avbildes til *C* måles vil man fra den eksisterende fordelingen kunne avgjøre om måleresultatet er ukjent.

Videre vil handlingseksemplene adaptivt tilpasse seg gradvise endringer i den optimale avbildning fordi de eldste handlingseksemplene slettes etter hvert som nye lagres.

Ulempen er at man risikere å miste viktige handlingseksempler hvis *n* er for lav i forhold til frekvensen av viktige måleresultat tilknyttet en gitt handling. Konsekvensen ved sletting av viktige handlingseksempler ser derimot kun ut til å være at man enten ved neste måleresultat vil behandle resultatet som ukjent eller at en annen mer sannsynlig handling avgjør utfallet (noe som på grunn av frekvensen til det glemte handlingseksempelet antagelig ville vært tilfellet uansett). Videre kan dette aspektet ved avbildningsmetoden justeres direkte og muligens adaptivt via *n*. Det ser altså ved første vurdering ut til at denne

ulempen er akseptabel i forhold til fordelene man oppnår med foreslått avbildningsmetode.

Under neste punkt foreslår vi en feildeteksjons- og feilkorrigeringsalgoritme basert på beskrevet avbildningsmetode.

4.3 Forslag til en feildeteksjons- og feilkorrigeringsalgoritme.

Algoritmen består av hovedprosedyren *UtførFeildeteksjon/feilkorrigering* og underprosedyren *VelgHandling*.

UtførFeildeteksjon/feilkorrigering koordinerer en feildeteksjons- eller feilkorrigeringsaksjon mot en gitt komponent eller tjeneste. Prosedyren benytter underprosedyren *VelgHandling* til å avgjøre hvilken handling som er best egnet for et gitt måleresultat. Først forsøker prosedyren å avgjøre en handling basert på lokale handlingseksempler. Hvis lokale handlingseksempler gir et tvilstilfellet eller måleresultatet er ukjent lokalt, benyttes *VelgHandling* til å avgjøre en handling globalt. På denne måten kombineres både komponent- og tjenestebestemt kunnskap med typebestemt kunnskap. Videre kan algoritmen og handlingseksemplene lett distribueres.

VelgHandling baserer seg på nærmeste-nabo metoder for å avgjøre om et måleresultat er ukjent, om det eksisterer tvil eller hvilken handling som er best egnet. Handlingseksemplene som benyttes ved en avgjørelse er knyttet til tjenestenavnet eller komponentnavnet gitt som innparameter til prosedyren.

Algoritme.

UtførFeildeteksjon/feilkorrigering(*Navn*)
{
    # Identifiserer tilgjengelige målefunksjoner.
    $M_1, ..., M_n \leftarrow MListe(Type(Navn))$;
    # Identifiserer tilgjengelige handlinger.
    $H_1, ..., H_m \leftarrow HListe(Type(Navn))$;
    # Plasserer resultatet av målefunksjonene utført mot *Navn* i en vektor.
    $\bar{x} \leftarrow [M_1(Navn), ..., M_n(Navn)]$;

    # Undersøker om lokale handlingseksempler kan benyttes til å avbilde $\bar{x}$.
    $LokalHandling \leftarrow VelgHandling(Navn, \bar{x})$;

    # Det samles inn handlingstemmer hvis $\bar{x}$ er ukjent lokalt eller det eksisterer tvil.
    Hvis *LokalHandling* == Ukjent eller Lokal*Handling* == Tvil så {
        # Teller opp stemmer for hver handling.
        For hvert listeelement *Navn'* i *NavneListe(Type(Navn))* gjør {
            $H \leftarrow VelgHandling(Navn', \bar{x})$;
            Hvis H != Ukjent og H != Tvil så {
                $Teller(H) \leftarrow Teller(H) + 1$;
                $Kvalifisert \leftarrow OK$;
            }
        }

        # Hvis ingen stemme ble gitt regner vi måleresultatet som ukjent lokalt.
        Hvis *Kvalifisert* != OK så {
            $LokalHandling \leftarrow Ukjent$;
        # Hvis handlingene fikk stemmer uten at vi fikk noen klar vinner regner vi måleresultatet som et tvilstilfelle.
        }ellers hvis $\forall h \in (1, ..., m)(\hat{p}(H_h | \bar{x}, [Teller(H_1), H_1],$ så {
                $..., [Teller(H_m), H_m]) < Tvil(Type(Navn)))$

            $LokalHandling \leftarrow Tvil$;
        # Ellers utføres handlingen som fikk flest stemmer.
        } ellers {

            $$LokalHandling \leftarrow \underset{h \in (H_1, ..., H_m)}{MaksArg}(Teller(h));$$

        }
    }
    # Utfører handling *LokalHandling* mot komponenten eller tjenesten med navn *Navn*. Returnerer navnet på vellykket handling hvis denne kan identifiseres.
    $r \leftarrow Handling(Navn, \bar{x}, LokalHandling)$;
    Hvis r != Ukjent og r != Tvil så {
        # Oppdaterer ringbufferet knyttet til komponenten eller tjenesten med navn *Navn* og den vellykkede handlingen r med måleresultatet $\bar{x}$.
        $Res(Navn, r) \leftarrow Res(Navn, r) \ll \bar{x}$;
    }
}

VelgHandling(*Navn*, $\bar{x}$)
{
    # Identifiserer tilgjengelige handlinger.
    $H_1, ..., H_m \leftarrow HListe(Type(Navn))$;

    # Returnerer Ukjent hvis måleresultatet er ukjent.
    Hvis $\hat{p}(\bar{x}|Res(Navn, H_1), ..., Res(Navn, H_m)) < Ukjent(Type(Navn))$ så {

        returner (Ukjent);

    # Returner Tvil hvis det er tvil mellom to eller flere handlinger.
    } ellers hvis $\forall h \in (1, ..., m)(\hat{p}(H_h|\bar{x}, [Res(Navn, H_1), H_1],$ så {
                $..., [Res(Navn, H_m), H_m]) < Tvil(Type(Navn)))$

        returner (Tvil);

    } ellers {
        # Returnerer handlingen som er den mest sannsynlige avbildningen til $\bar{x}$ gitt handlingseksemplene knyttet til komponenten eller tjenesten med navn *Navn*.

$$H \leftarrow \underset{h \in (H_1, ..., H_m)}{MaksArg}(\hat{p}(h|\bar{x}, [Res(Navn, H_1), H_1], ..., [Res(Navn, H_m), H_m]));$$

        returner (*H*);
    }
}

Vi beskriver her først hver av prosedyrene. Deretter beskrives sannsynlighetsestimatene benyttet i *UtførFeildeteksjon/feilkorrigering* og *VelgHandling*.

UtførFeildeteksjon/feilkorrigering.

UtførFeildeteksjon/feilkorrigering kalles med navnet på en komponent eller en tjeneste. Prosedyren utfører først målingene bestemt av komponent- eller tjenestetypen, *MListe(Type(Navn))*. Resultatet av målingene tilordnes vektoren $\bar{x}$.

Deretter benyttes prosedyren *VelgHandling* til å avbilde $\bar{x}$ til en handling med hensyn på de lokale handlingseksemplene.

Hvis måleresultatet $\bar{x}$ er ukjent lokalt eller gir et tvilstilfelle lokalt, bestemmes handlingen globalt ved å samle inn stemmer som vist i prosedyren. Handlingen med flest stemmer utføres ved klar "seier". Ellers rapporteres ukjent måleresultat eller tvilstilfelle.

Etter utførelse av handling, rapportering av ukjent måleresultat eller rapportering av tvilstilfelle har vi en returverdi *r* fra prosedyren *Handling*. Hvis returverdien er navnet på en vellykket handling har vi et nytt handlingseksempel fra $\bar{x}$ til *r*. Handlingens ringbuffer *Res(Navn, r)* knyttet til komponenten eller tjenesten med navn *Navn* oppdateres da med måleresultatet $\bar{x}$.

VelgHandling.

I *VelgHandling* er det meningen at det først beregnes sannsynligheten for at måleresultatet $\bar{x}$ er fra fordelingen av måleresultat inneholdt i handlingseksemplene knyttet til komponenten eller tjenesten med navn *Navn*. Litt forenklet kan dette beregnes ved $\hat{p}(\bar{x}|Res(Navn, H_1), ..., Res(Navn, H_m))$. *Res(Navn, H_i)* betegner de *n* siste måleresultatene knyttet til handling H_i mot komponenten eller tjenesten med navn *Navn*.

Hvis denne verdien er lavere enn en gitt terskel, *Ukjent(Type(Navn))*, returneres symbolet Ukjent. Det betyr at måleresultatet med en viss sansynlighet ikke tilhører fordelingen av måleresultat inneholdt i handlingseksemplene.

Hvis a posteriori sannsynligheten for hver handling *h* gitt måleresultat $\bar{x}$ og handlingseksemplene knyttet til hver handling, $\hat{p}(h|\bar{x}, [Res(Navn, H_1), H_1], ..., [Res(Navn, H_m), H_m])$, er mindre enn terskelen for tvil, *Tvil(Type(Navn))*, returneres symbolet Tvil.

Hvis måleresultatet ikke er ukjent og ikke gir et tvilstilfelle, identifiseres handlingen med høyest a posteriori sannsynlighet gitt måleresultat $\bar{x}$ og handlingseksemplene knyttet til hver handling, $\hat{p}(h|\bar{x}, [Res(Navn, H_1), H_1], ..., [Res(Navn, H_m), H_m])$. Navnet på denne handlingen returneres.

Sannsynlighetsestimat.

Her skisserer vi kort hvordan de ulike sannsynlighetsestimatene kan beregnes.

For å beregne $\hat{p}(H_h|\bar{x}, [Teller(H_1), H_1], ..., [Teller(H_m), H_m]) < Tvil(Type(Navn))$ kan vi dividere antall stemmer gitt til handling H_h med antall stemmer gitt totalt.

For å beregne $\hat{p}(\bar{x}|Res(Navn, H_1), ..., Res(Navn, H_m))$ kan vi benytte et k_n-nærmeste-nabo estimat (se [4] for en utledning)

$$\hat{p}_n(\bar{x}|Res(Navn, H_1), ..., Res(Navn, H_m)) = \frac{k_n/n}{V_n}.$$

Her er n antall lagrede måleresultat for handlingen med færrest handlingseksempler, $k_n = \sqrt{n}$ og V_n er volumet til en hyperkube som inneholder de k_n nærmeste naboene til $\bar{x}$. Dette gir en middlet tetthetsfordeling hvis nøyaktighet er avhengig av *n*. Vi velger å la *n* bestemmes av antall lagrede måleresultat for handlingen med færrest handlingseksempler fordi man da er sikker på å ikke trekke inn for mange naboer.

Sannsynligheten $\hat{p}(h|\bar{x}, [Res(Navn, H_1), H_1], ..., [Res(Navn, H_m), H_m])$ for en handling *h* beregnes også ved et k_n-nærmeste-nabo estimat, men denne gangen for *a posteriori* sannsynligheten (se [4] for utledning og egenskaper):

$$\hat{p}_n(h|\bar{x}, [Res(Navn, H_1), H_1], ..., [Res(Navn, H_m), H_m]) = \frac{k_h}{k_n}.$$

Her er n er antall lagrede måleresultat for handlingen med færrest handlingseksempler, $k_n = \sqrt{n}$ og k_h er antall måleresultat knyttet til handlingen *h* av de k_n nærmeste måleresultatene blant handlingseksemplene.

4.4 Konklusjon.

Vi har foreslått en adaptiv avbildningsmetode for bruk til feildeteksjon og feilkorrigering. Denne metoden er basert på nærmeste-nabo metodene. Nærmeste-nabo metodene gir generellt gode approksimasjoner og danner grunnlaget for flere vellykkede feilhåndteringssystemer deriblant to beskrevet i [9] og [10]. Foreslått avbildningsmetode skiller seg fra de beskrevne ved at man eksplisitt kan definere en terskel for ukjente, antall lagrede handlingseksempler og terskel for tvil eksplisitt.

Videre har vi foreslått en automatisk og adaptiv feildeteksjons- og feilkorrigeringsalgoritme som interakterer med et kommunikasjonssystem via feildeteksjons- og feilkorrigeringsdatabasen definert i *del 3*. Denne algoritmen baserer seg i første omgang på

foreslått avbildningsmetode. Hvis det viser seg at foreslått avbildningsmetode ikke har ventede egenskaper kan vi falle tilbake på de allerede velutprøvde men potensielt ikke like egnede feilhåndteringsmetoder basert på “case”-basert resonnering lik metodene beskrevet i [9] eller [10].

5 Svakheter og usikkerhetsmomenter ved foreslått feildeteksjons- og feilkorrigeringsmetode.

I foregående deler har vi skissert et forenklet bilde av feildeteksjon og feilkorrigering i kommunikasjonssystemer. Dette for å muliggjøre automatisering av enkle feildeteksjons- og feilkorrigeringsoppgaver. Følgende vanskelighet har for oss gjort automatisering av feildeteksjon og spesielt feilkorrigering til en spennende og utfordrende oppgave.

Under arbeidet med Telenors TCP/IP-SNA gateway systemer har vi erfart at enkelte feil kun kan korrigeres ved kompleks og sammensatt omkonfigurering av flere komponenter. Antall kombinasjoner av komponenter som kan inngå i en slik omkonfigurering og den respektive kombinatoriske veksten av konfigurasjonsmuligheter gjør denne typen feilkorrigering svært vanskelig.

Vi griper ikke denne automatiseringsutfordringen i dette kapittelet men tenker oss at feildeteksjons- og feilkorrigeringsmetoden vi foreslår skal benyttes i feilsituasjoner med enkle lokale feil hvor rutinemessige håndgrep korrigerer feilen. Dette kan tenkes å være omstart av en komponent eller valg av en egnet komponent- eller tjenestekonfigurasjon fra et lite antall predefinerte konfigurasjoner.

Usikkerhetsmomentet ved automatisering av rutinemessig feildeteksjon og feilkorrigering er følgende: det er ikke sikkert at beslutningsgrunnlaget man får ved å automatisk utføre målinger lokalt i en komponent eller tjeneste er tilstrekkelig til å bestemme handlinger. Vi mener derimot at det er god grunn til å tro at beslutningsgrunnlaget er tilstrekkelig da både menneskelige systemadministratorer og systemeksperter i stor grad utfører handlinger i kommunikasjonssystemet på grunnlag av måleresultat.

En svakhet ved foreslått metode er at den kun foreslår en handling i stedet for å rangere handlingsalternativer. Sistnevnte fremgangsmåte er mer hensiktsmessig hvis handlingene rapporteres som forslag til for eksempel en systemadministrator eller systemekspert eller hvis man ønsker at handlinger skal foretas automatisk og sekvensielt i en prioritert rekkefølge frem til feilen er korrigert.

Vi konkluderer med at ved skissert kompleksitet og usikkerhet vil trolig en implementasjon og uttesting av foreslått metode avdekke behov for endringer og utvidelser av metoden. Vi regner derfor forslaget i dette kapittelet som summen av de erfaringer vi har gjort med automatisering av feildeteksjon og feilkorrigering i enkle situasjoner og som et forenklet grunnlag for drøftelser rundt det distribuerte feilhåndteringssystemet som foreslås i *kapittel 4*.

6 Oppsummering.

Vi har i dette kapittelet definert et språk for spesifikasjon av feildeteksjons- og feilkorrigeringsgrensesnitt. Videre har vi foreslått et oversettelseskjema for automatisk generering av en feildeteksjons- og feilkorrigeringdatabase fra en streng i spesifikasjonsspråket. Det er tenkt at spesifikasjonsspråket og feildeteksjons-/feilkorrigeringsdatabasen skal danne et uniformt rammeverk for feildeteksjon- og feilkorrigering.

Videre har vi drøftet ulike approksimative avbildningsmetoder for bruk til automatisk feildeteksjon og feilkorrigering. Disse avbildningsmetodene ser ikke ut til å være optimale med hensyn på problemene vi fremhevet. Det ble derfor foreslått en avbildningsmetode

hvis parametre direkte er relatert til disse problemene. Til slutt baserte vi en feildeteksjons- og feilkorrigeringsalgoritme på denne metoden.

Innenfor rammen av hovedoppgaven har det ikke vært anledning til å implementere og utføre eksperimenter med foreslått metode. Det har derimot blitt drøftet uhåndterte utfordringer og mulige usikkerhetsmomenter ved metoden.

Vi viser til slutt til at flere sentraliserte "case"-baserte resonneringssystemer (hvor nærmeste-nabo metoder er grunnleggende) har vist seg å være vellykkede til feildeteksjon og feilkorrigering i mer sammensatte situasjoner enn de lokale vi håndterer. Vi håper at grunnprinsippene i vårt forslag kan supplere disse fremgangsmåtene i tilfeller hvor de problemene vi har fremhevet i *del 1* er fremtredende.

7 Referanser.

[1] A. V. Aho, R. Sethi & J. D. Ullman, Compilers: Principles, Techniques and Tools, Addison-Wesley Publishing Company, 1986, s. 26-32.
[2] A. V. Aho, R. Sethi & J. D. Ullman, Compilers: Principles, Techniques and Tools, Addison-Wesley Publishing Company, 1986, s. 159-340.
[3] Duda & Hart, Pattern Classification and Scene Analysis, John Wiley & Sons. 1973, s. 44-80.
[4] Duda & Hart, Pattern Classification and Scene Analysis, John Wiley & Sons, 1973, s. 95-98.
[5] B. D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, 1996, s. 17-173.
[6] B. D. Ripley, Pattern Recognition and Neural Networks, Cambridge Univer
sity Press, 1996, s. 181-207.
[7] U. Black, Network Management Standards: The OSI, SNMP and CMOL Protocols, 1992, McGraw-Hill, Inc.
[8] Rober D. Gardner og David A. Harle, Alarm Correlation and Network Fault Resolution using Kohonen Self-Organising Map, GLOBECOM 97, s. 1398-1402.
[9] Yuh Foong David Law, Sew Bun Foong og Shee Eng Jeremiah Kwan, An integrated Case-Based Reasoning Approach for Intelligent Help Desk Fault Management.
[10] Sue Abu-Hakima, The DPN Expert Advisor: Canadian AI Success Story #3, Canadian Artificial Intelligence, 1994, nr 33, s.14-19.
[11] Jill Huntington-Lee, Kornel Terplan og Jeff Gibson, HP OpenView, A Manager's Guide, 1997.
[12] B. D. Ripley, Pattern Recognition and Neural Networks, Cambridge Univer
sity Press, 1996.

Kapittel 4: Et forslag til et automatisk, distribuert og adaptivt feilhåndteringssystem.

Ole-Christoffer Granmo,

Institutt for Informatikk,

Universitetet i Oslo.

10/2/99

1 Innledning.

Etter hvert som kommunikasjonssystemer blir større og mer komplekse kommer hovedsvakheten ved sentralisert feilhåndtering klart frem; kommunikasjonsressursene og prosesseringsressursene begrenser hvor mye data som kan overføres og behandles sentralt. Det oppstår dermed en konflikt mellom behovet for feilhåndtering i store kommunikasjonssystemer på den ene siden og behovet for høy feilhåndteringspresisjon på den andre siden [1]. For å bøte på dette er flere distribuerte feilhåndteringssystemer og feilhåndteringsmodeller foreslått [1, 2]. En utfordring er å oppnå automatisk og global feilhåndtering på tross av distribusjonen. En annen utfordring er å oppnå adaptiv feilhåndtering med hensyn på endringer i det underliggende kommunikasjonssystemet.

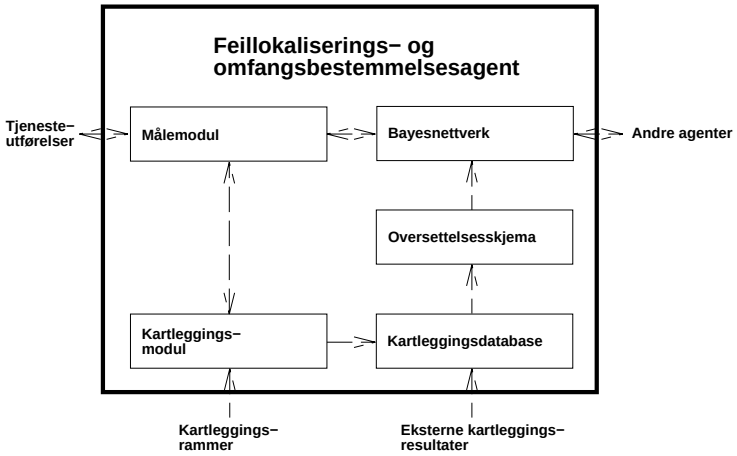
I dette kapittelet foreslår vi et automatisk, distribuert og adaptivt feilhåndteringssystem for global håndtering av feil. Da feillokaliserings- og omfangsbestemmelsesmetoden fra *kapittel 2* er grundigst behandlet i hovedoppgaven vil vi i stor grad fokusere på mulige utvidelser av denne metoden i dette kapittelet. Videre begrenser vi oss til feilsituasjoner hvor feil kan korrigeres ved en omstart eller ved enkel omkonfigurering basert på valg av konfigurasjon fra et gitt antall predefinerte konfigurasjoner (se *kapittel 3*).

Vi tenker oss at feilhåndteringssystemet skal bestå av to typer *agenter*; feillokaliserings- og omfangsbestemmelsesagenter basert på metoden foreslått i *kapittel 2* og feildeteksjons- og feilkorrigeringsagenter basert på metoden foreslått i *kapittel 3*. Med en agent menes en enhet som iakttar omgivelsene ved hjelp av sensorer og utfører handlinger i omgivelsene ved hjelp av effektorer [6].

I *del 2* og *del 3* beskrives hver av de to agenttypene. Distribuerte instanser av disse skal samarbeide om å oppnå global feilhåndtering. Det er to grunnleggende former for samarbeid å velge mellom; hierarkisk kontroll eller likestilte samarbeid [3]. I *del 4* drøftes avveiningen som må gjøres mellom disse alternativene. Ved begge alternativer kan endringer i underliggende kommunikasjonssystem føre til utdatering av samarbeid. Vi foreslår derfor i *del 5* hvordan vi tenker oss at feilhåndteringssystemet automatisk og adaptivt kan tilpasse seg et gitt kommunikasjonssystem. Deretter i *del 6* beskrives detaljene i hvordan operasjonen og interaksjonen mellom inngående agenter kan foregå. Frem til *del 7* velger vi å presentere en forenkelt modell av feilhåndtering. Dette for å gi et klart bilde av valgene som gjøres. Da foreslått feilhåndteringssystem ikke er implementert eller testet ut konkluderer vi i *del 7* med å beskreve de underliggende utfordringene ved automatisk feilhåndtering samt begrensningene og usikkerhetsmomentene ved foreslått system.

2 Feillokaliserings- og omfangsbestemmelsesagentene.

Vi foreslår at en feillokaliserings- og omfangsbestemmelsesagent i feilhåndteringssystemet består av en kartleggingsmodul, en kartleggingsdatabase, et oversettelsesskjema, et adaptivt generert Bayesnettverk og en målemodul. Sammordingen av modulene illustreres i *figur 1*.



Figur 1: *Samordningen av moduler i en feillokaliserings- og omfangsbestemmelsesagent.*

Vi lar målet til en feillokaliserings- og omfangsbestemmelsesagent være å danne et globalt bilde av tilstanden i underliggende kommunikasjonssystem med henblikk på styring av feilkorrigeringen.

Kartleggingsmodulen kartlegger med jevne mellomrom sammensetningen til tjenestene i kommunikasjonssystemet innenfor gitte kartleggingsrammer. Kartleggingen utføres ved hjelp av målemodulen. Spesifikasjonsspråket fra *kapittel 2* danner formatet på kartleggingsresultat.

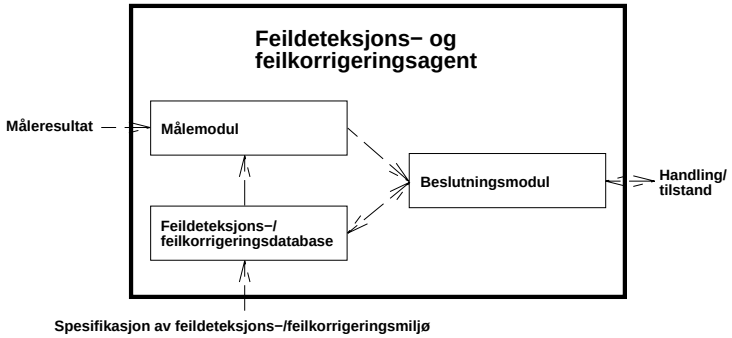
Resultatet av kartleggingen lagres i kartleggingsdatabasen sammen med *eksterne kartleggingsresultater*. Med eksterne kartleggingsresultater menes kartleggingsresultater fra andre feillokaliserings- og omfangsbestemmelsesagenter eller en ekstern bruker.

Oversettelsesskjemaet med tilhørende maler (se *kapittel 2*) benyttes så til å holde et Bayesnettverk oppdatert ved endringer i kartleggingsdatabasen. Bayesnettverket modellerer årsakssammenhenger mellom tilstanden til tjenestene, aktiveringssettene, komponentsamarbeidene og komponentene som er lagret i kartleggingsdatabasen.

Agenten danner et bilde av tilstanden i underliggende kommunikasjonssystem ved å estimere de mest sannsynlige tilstandene i Bayesnettverket gitt tilstander bestemt av målemodulen eller av andre agenter. På denne måten ser vi for oss at lokasjonen og omfanget til feil kan bestemmes. I neste del gis en kort beskrivelse av hvordan vi tenker oss feildeteksjons- og feilkorrigeringsagentene.

3 Feildeteksjons- og feilkorrigeringsagentene.

Vi foreslår at feildeteksjonsagenter og feilkorrigeringsagenter i feilhåndteringssystemet består av en målemodul, en feildeteksjons-/feilkorrigeringsdatabase og en beslutningsmodul. Samordningen av modulene illustreres i *figur 2*.



Figur 2: *Samordningen av moduler i en feildeteksjons- og feilkorrigeringsagent.*

Vi lar målet til en feildeteksjonsagent være å bestemme tilstanden til en tilknyttet komponent ved å utføre målinger direkte i komponenten. Målet til en feilkorrigeringsagent lar vi være å på tilsvarende måte bestemme en egnet korrigerende handling ved feilsituasjoner av bestående av den typen feil vi begrenser oss til.

Når en feildeteksjonsagent eller en feilkorrigeringsagent aktiveres utfører målemodulen målinger i tilknyttet komponent. Dette for å danne et bilde av tilstanden i komponenten. De målingene som skal utføres er registrert i feildeteksjons-/feilkorrigeringsdatabasen.

Videre inneholder feildeteksjons-/feilkorrgieringsdatabasen handlingseksempler, dvs tidligere måleresultat knyttet til en da passende handling.

Beslutningsmodulen er basert på feildeteksjons- og feilkorrigeringsalgoritmen foreslått i *kapittel 3*. Kort beskrevet bestemmer beslutningsmodulen en handling ved å samordne det nye måleresultatet med handlingseksemplene i feildeteksjons-/feilkorrigeringsdatabasen. Hvis feildeteksjons-/feilkorrigeringsdatabasen ikke inneholder egnede handlingseksempler ber beslutningsmodulen om råd. Først ved å overføre måleresultatet til andre feildeteksjons- og feilkorrigeringsagenter som håndterer komponenter av samme type. Deretter hvis ingen av agentene har sikre forslag, overføres måleresultatet til en menneskelig systemadministrator eller systemekspert.

Hvis det nye måleresultatet med stor sikkerhet kan kobles til en gitt handling, f.eks. ved ekstern veiledning, lagrer beslutningsmodulen denne koblingen som et handlingseksempel i feildeteksjons-/feilkorrigeringsdatabasen.

På denne måten tenker vi oss at feildeteksjonsagenter og feilkorrigeringsagenter adaptivt kan lære å bestemme enkle komponenttilstander og bestemme enkle korrigerende handlinger ved feil av de typene vi begrenser oss til. Se *kapittel 3* for en nærmere beskrivelse av denne prosessen.

I neste del drøftes hvordan samarbeid mellom distribuerte instanser av beskrevne agenttyper kan organiseres.

4 Hierarkisk kontroll vs. likestilt samarbeid.

Ved sentralisert nettverksadministrasjon generelt og sentralisert feilhåndtering spesielt, får man som nevnt på grunn av kommunikasjons- og prosesseringsbegrensninger en konflikt mellom behovet for feilhåndtering i store kommunikasjonssystemer på den ene siden og behovet for høy feilhåndteringspresisjon på den andre siden [1]. Begrensningen kan omgås ved å distribuere feilhåndteringen.

Feilhåndteringen kan distribueres ved å dele den opp etter abstraksjonsnivå, organisasjonsmessige grenser eller ulike feilhåndteringsoppgaver i kommunikasjonssystemet. Hver del kan da administreres av en agent.

For å oppnå feilhåndtering på et globalt plan på tross av distribusjonen må agentene samarbeide. Samarbeid mellom agentene kan være likestilt, organiseres i et kontrollhierarki eller danne et hybrid av disse tilnærmingene [3]. Likestilte agenter inngår samarbeid om å oppnå egne og felles mål etter behov. Når en agent ikke er i stand til å nå et mål alene, vil den forsøke å finne andre agenter som kan hjelpe den i å nå målet. I et kontrollhierarki foregår feilhåndteringen ovenfra og ned etter fastlagte mønstre. På høyere nivå foregår tidkrevende, abstrakt og global feilhåndtering, mens det på lavere nivå foregår hurtig, detaljert og lokal feilhåndtering. En agent når sine mål selv, eller ved å fordele delmål til agenter lenger ned i kontrollhierarkiet.

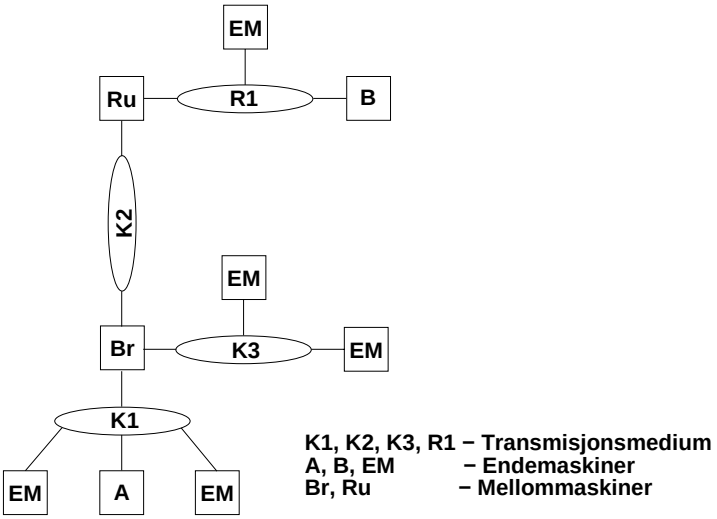
En ulempe ved hierarkisk kontroll av feilhåndteringen synes å være at kontrollhierarkiet bindes til den strukturelle sammensetningen i kommunikasjonssystemet. Det kan dermed bli stivbent med hensyn på strukturelle forandringer. Fordelen er at man får oversiktlig og forutsigbar feilhåndtering. Likestilte samarbeid er riktig nok mer fleksible og mer robuste mot strukturelle forandringer, men feilhåndteringen blir lettere uoversiktlig og uforutsigbar når underliggende kommunikasjonssystem er store og komplekse.

De fleste kommunikasjonssystemer er hierarkisk oppbygd. Det er dermed naturlig å velge hierarkisk kontroll av feilhåndteringen, organisert etter abstraksjonslag i kommunikasjonssystemet. Vi ser foreløpig bort fra de andre oppdelingsalternativene, dels fordi de delvis er sammenfallende med valgt oppdeling og dels fordi de vil kreve mer fleksible og mer innfløkte agentsamarbeid enn det et rent kontrollhierarki kan tilby.

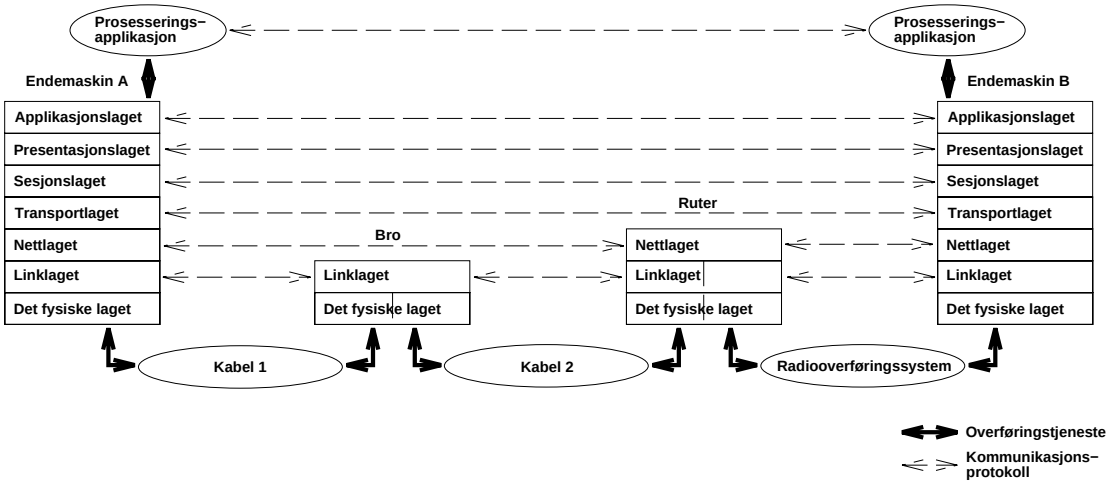
For å bøte på den beskrevne ulempen ved hierarkisk kontroll foreslår vi i neste del hvordan agentkontrollhierarkiet vårt kan bygges opp adaptivt med hensyn på den strukturelle sammensetningen til et kommunikasjonssystem.

5 Adaptiv oppbygging av agentkontrollhierarkiet.

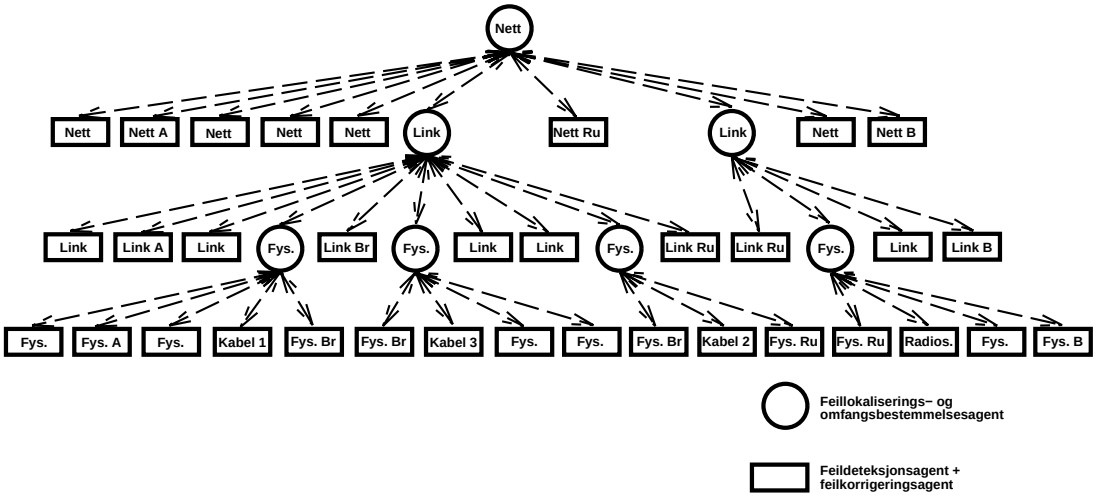
Som begrunnet i *del 4* foreslår vi å distribuere feilhåndteringssystemet i et kontrollhierarki etter abstraksjonslag i kommunikasjonsystemet. For å lettere adaptere kontrollhierarkiet til et gitt kommunikasjonssystem, åpner vi videre for felles "undersåtter". Kontrollhierarkiet gir dermed en rettet asyklisk graf. Vi lar en *intern node* i grafen representere en feillokaliserings- og omfangsbestemmelsesagent, mens en *bladnode* representerer en feildeteksjonsagent og en feilkorrigeringsagent. Med en intern node menes en node med etterfølgere i grafen. Med en bladnode menes en node uten etterfølgere. *Figur 3* illustrerer et kontrollhierarki av denne typen for et gitt kommunikasjonssystem. Dette kommunikasjonssystemet illustreres i *figur 1* og *figur 3* fra *kapittel 1*. Disse figurene gjengis også her.



Kapittel 1, figur 1: *Endemaskiner knyttet sammen via transmisjonsmedier og mellommaskiner.*



Kapittel 1, figur 3: *Et eksempel på kommunikasjon mellom applikasjoner.*



Figur 3: *Et kontrollhierarki for feilhåndtering basert på eksempelkommunikasjonssystemet fra kapittel 1.*

I *figur 3* refererer navnet *Nett* til nettlaget og navnet *Link* til linklaget i eksempelkommunikasjonssystemet. Feillokaliserings- og omfangsbestemmelsesagenten på nettnivå kontrollerer feildeteksjonsagentene og feilkorrigeringsagentene på samme nivå. Vi tenker oss at hver feildeteksjonsagent og feilkorrigeringsagent på nettnivå håndterer hvert sitt nettlag i maskinene fra *figur 1* i *kapittel 1*. Videre kontrollerer agenten feillokaliserings- og omfangsbestemmelsesagentene på linknivå. Disse feillokaliseringsagentene håndterer hvert sitt subkommunikasjonssystem fra *figur 1* i *kapittel 1* (hvor det ene subkommunikasjonssystemet består av kablene *K1*, *K2* og *K3* med tilknyttede maskiner mens det andre subkommunikasjonssystemet består av radioverføringssystemet *R1* med tilknyttede maskiner). Hver av feillokaliserings- og omfangsbestemmelsesagentene på linknivå kontrollerer igjen feildeteksjonsagenter og feilkorrigeringsagenter på eget nivå samt feillokaliserings- og omfangsbestemmelsesagenter på fysisk nivå. Feildeteksjonsagentene og feilkorrigeringsagentene på linknivå tenker vi oss videre at håndterer linklaget i hver sin maskin. På denne måten distribueres feilhåndteringen i et kontrollhierarki etter abstraksjonslag i kommunikasjonssystemet.

Under avsnitt *5.1* forslår vi hvordan et slikt agentkontrollhierarki kan bygges opp med hensyn på den strukturelle sammensetningen til et kommunikasjonssystem. Deretter under *5.2* beskriver vi hvordan vi tenker oss at beregningene i Bayesnettverk-modulen til hver feillokaliserings- og omfangsbestemmelsesagent kan samordnes. Det gis en oppsummering av denne delen under *5.3*.

5.1 Oppbyggingsprosessen.

Under dette punktet definerer vi et oversettelsesskjema som bygger opp et agentkontrollhierarki rekursivt. Definisjonen er basert på oversettelsesskjemaet fra *kapittel 2*. Dette oversettelseskjemaet genererer et Bayesnettverk for tjenesteorientert feillokalisering og omfangsbestemmelse fra sammensetningen til tjenester. Det nye oversettelsesskjemaet skal i tillegg dele opp Bayesnettverket i moduler som danner kjernen i distribuerte feillokaliserings- og omfangsbestemmelsesagenter. Videre skal det administrere feildeteksjons- og feilkorrigeringsagentene i hierarkiet.

Vi tenker oss at resulterende Bayesnettverk-moduler sammen skal danne et globalt bilde av tilstanden i kommunikasjonssystemet. Dermed kan feilhåndteringen utføres automatisk og distribuert men på et globalt plan.

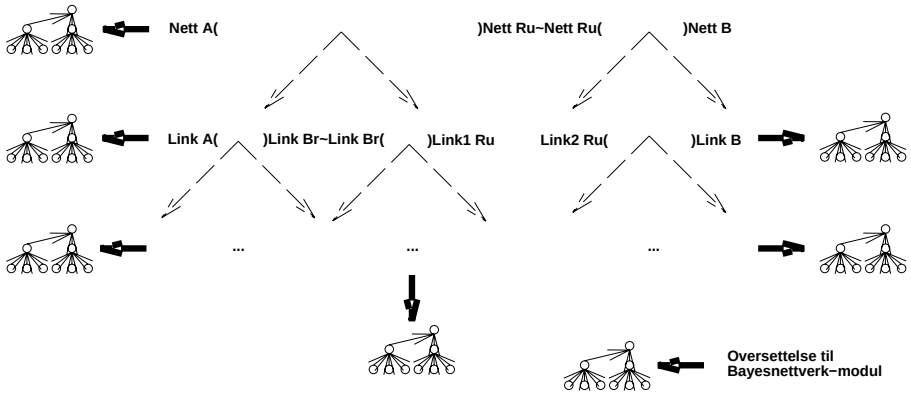
Først ser vi på hvordan strenger i spesifikasjonsspråket fra *kapittel 2* kan splittes opp etter abstraksjonsnivåene i spesifikasjonen. Deretter beskrives hvordan en agent kan identifisere subkommunikasjonssystemene på et gitt abstraksjonsnivå. Til slutt defineres hvordan agentkontrollhierarkiet bygges opp adaptivt ved hjelp av et oversettelsesskjema lokalisert i hver feillokaliserings- og omfangsbestemmelsesagent.

Oppsplitting av spesifikasjon.

Oversettelsesskjemaet fra *kapittel 2* genererer et Bayesnettverk fra spesifikasjonen av sammensetningen til tjenester i et gitt kommunikasjonssystem. Her ønsker vi å skille de ulike abstraksjonsnivåene i en spesifikasjon fra hverandre slik at genereringen av Bayesnettverket kan splittes opp i et abstraksjonshierarki. Hver Bayesnettverk-modul i abstraksjonshierarkiet danner kjernen i en feillokaliserings- og omfangsbestemmelsesagent. *Figur 4* illustrerer denne oppsplittingen med følgende streng i spesifikasjonsspråket fra *kapittel 2*:

Nett A(Link A(...)Link Br~Link Br(...)Link1 Ru)Nett Ru~Nett Ru(Link2 Ru(...)Link B)Nett B.

Strengen spesifiserer sammensetningen til en tjeneste basert på *figur 3* fra *kapittel 1*. Fysisk nivå er abstrahert bort.



Figur 4: *Oppsplitting av Bayesnettverk i et abstraksjonshierarki.*

Leseren bør merke seg sammenhengen mellom *figur 3* og *figur 4*. På nettnivå oversettes nettdelen av spesifikasjonen til en Bayesnettverk-modul. Videre skilles spesifikasjonen av tjenester på lavere abstraksjonsnivå ut. Linkdelen av disse oversettes igjen til hver sin Bayesnettverk-modul. Deretter fortsetter oversettelsen på fysisk nivå.

På denne måten splittes Bayesnettverket opp i et abstraksjonshierarki.

Identifisering av subkommunikasjonssystemer fra spesifikasjon.

Med den oppsplittingen som er beskrevet under forrige avsnitt får vi et eller flere *subkommunikasjonssystemer* på hvert abstraksjonsnivå. Med et subkommunikasjonssystem menes her en gruppering av alle tjenester med overlappende komponentsammensetning på et gitt abstraksjonsnivå. Vi ønsker at en feillokaliserings- og omfangsbestemmelsesagent skal plassere ut en ny feillokaliserings- og omfangsbestemmelsesagent i hvert underliggende subkommunikasjonssystem. Videre ønsker vi at de utplasserte agentene skal fortsette oversettelsen på sitt nivå. Det betyr at den utplasserende feillokaliserings- og omfangsbestemmelsesagenten må identifisere hvert subkommunikasjonssystem og spesifikasjonen av sammensetningen til tjenester innenfor disse.

Man kan avgjøre om to tjenester tilhører samme subkommunikasjonssystem ved å definere en ekvivalensrelasjon *FKS* over sett, hvor et sett inneholder komponentene som aktiveres under en gitt tjeneste på et gitt abstraksjonsnivå:

$$FKS(s, t) = s \cap t \neq \varnothing .$$

To komponentsett *s* og *t* befinner seg innenfor et felles subkommunikasjonssystem hvis de har en felles komponent, eller hvis det eksisterer en kjede mellom *s* og *t* av komponentsett som parvis oppfyller kravet om felles komponent.

Tjenester hvis komponentsett danner en ekvivalensklasse rammer inn et subkommunikasjonssystem.

Adaptiv oppbygging av agentkontrollhierarkiet.

Vi tenker oss at agentkontrollhierarkiet bygges opp ved å plassere en feillokaliserings- og omfangsbestemmelsesagent i kommunikasjonssystemet. Denne agenten kaller vi rotagenten i feilhåndteringssystemet. Rotagenten gis kartleggingsrammer som begrenser kartleggingen oppad i abstraksjonshierarkiet til et gitt nivå, f. eks. nettnivået i OSI-referansemodellen. Videre spesifiseres sammensetningen til tjenestene agenten skal overvåke for ekstern bruker.

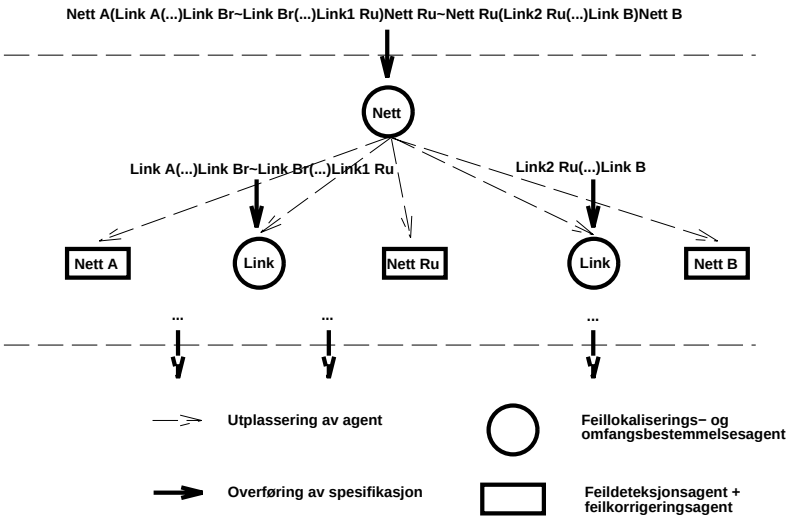
En utplassert feillokaliserings- og omfangsbestemmelsesagent kartlegger med jevne mellomrom sammensetningen til tjenestene i kommunikasjonssystemet fra sin lokasjon innenfor gitte kartleggingsrammer. Resultatet av kartleggingen lagres i kartleggingsdatabasen sammen med eksterne kartleggingsresultater.

Ved endringer i kartleggingsdatabasen oppdateres Bayesnettverk-modulen og barna i kontrollhierarkiet ved hjelp av et oversettelsesskjema. Oversettelseskjemaet genererer et Bayesnettverk som modellerer årsakssammenhenger mellom tilstanden til tjenestene, aktiveringssettene, komponentsamarbeidene og komponentene på agentens abstraksjonsnivå. Videre utskilles spesifikasjonen av sammensetningen til tjenester på lavere abstraksjonsnivå.

For hvert identifisert subkommunikasjonssystem på underliggende abstraksjonsnivå utplasseres en feillokaliserings- og omfangsbestemmelsesagent hvis en agent ikke allerede er utplassert. Hver utplassert agent gis kartleggingsrammer som begrenser kartleggingen oppad til underliggende abstraksjonsnivå samt kartleggingsrammer som begrenser bredden på kartleggingen til gitt subkommunikasjonssystem. Utskilte spesifikasjoner overføres så til korresponderende utplasserte agenter. Dermed kan de utplasserte feillokaliserings- og omfangsbestemmelsesagentene fortsette kartleggingen og oversettelsen på sitt abstraksjonsnivå.

Til slutt utplasseres en feildeteksjonsagent og en feilkorrigeringsagent for hver komponent på feillokaliserings- og omfangsbestemmelsesagentens abstraksjonsnivå hvis agentene ikke allerede er utplassert.

Denne prosessen illustreres i *figur 5*.



Figur 5: *Oppbygging av kontrollhierarki fra spesifikasjon av tjeneste.*

Figuren illustrerer at eksempelstrengen gis som et eksternt kartleggingsresultat til en eksisterende feillokaliserings- og omfangsbestemmelsesagent på nettnivå. Denne agenten oversetter strengen ved å:

1. Oppdatere Bayesnettverk-modulen i agenten med nettdelen av spesifikasjonen.
2. Plassere ut en feillokaliserings- og omfangsbestemmelsesagent for hvert identifisert subkommunikasjonssystem på linknivå.
3. Overfører utskilte spesifikasjoner av tjenester på lavere abstraksjonsnivå til korresponderende feillokaliserings- og omfangsbestemmelsesagenter på linknivå.
4. Plassere ut en feildeteksjonsagent og en feilkorrigeringsagent for hver identifisert komponent på nettnivå.

Det modifiserte oversettelsesskjemaet som utfører denne prosessen i hver feillokaliserings- og omfangsbestemmelsesagent defineres i *tabell 1* på tilsvarende måte som i *kapittel 2*.

Produksjon	**Semantiske regler**
Kommunikasjonssystem -> Tjenesteliste	*Tjenesteliste.nivå* := *0*; > OppdaterFeillokaliserings-/omfangsbestemmelsesagenter(); FjernOverflødigeAgenter();
Tjenesteliste -> Tjeneste **;** *Tjenesteliste'*	*Tjeneste.nivå* := *Tjenesteliste.nivå*; *Tjenesteliste'.nivå* := *Tjenesteliste.nivå*; >
Tjenesteliste -> Tjeneste **;**	*Tjeneste.nivå* := *Tjesteliste.nivå*; >

K -> ***Komp* : *Type***	Hvis *K.nivå* == *0* så {
	P*Type*(*Komp*);
	
	P*Type*Deteksjon(*Komp*Deteksjon | *Komp*);
	P*Type*Korrigering(*Komp* | *Komp*Korrigering);
	
	OppdaterFeildeteksjonsagent(***Type***, ***Komp***);
	OppdaterFeilkorrigeringsagent(***Type***, ***Komp***);
	}
	K := ***Komp***;

K -> ***Komp* : *Type* (** *Tjeneste* **) *Komp'***	*Tjeneste.nivå* := *K.nivå + 1*; > Hvis *K.nivå* == *0* så { **P*Type*(*Komp*)**; **P*Type*(*Komp'*)**; **P*Type*rel(*Komp*()*Komp'*	*Komp*, *Tjeneste*, *Komp'*)**; **P*Type*Deteksjon(*Komp*Deteksjon | *Komp*)**; **P*Type*Korrigering(*Komp* | *Komp*Korrigering)**; **P*Type*Deteksjon(*Komp'*Deteksjon | *Komp'*)**; **P*Type*Korrigering(*Komp'* | *Komp'*Korrigering)**; OppdaterEkvivalensklasser(*Tjeneste)*; OppdaterFeildeteksjonsagent(***Type***, ***Komp***); OppdaterFeilkorrigeringsagent(***Type***, ***Komp***); OppdaterFeildeteksjonsagent(***Type***, ***Komp'***); OppdaterFeilkorrigeringsagent(***Type***, ***Komp'***); } *K* := ***Komp* () *Komp'***;

Tjeneste -> K
~ Tjeneste'
	K.nivå := *Tjeneste.nivå*;
	Tjeneste'.nivå := *Tjeneste.nivå*;
	>
	Hvis *K.nivå* == *0* så {
	P*og*(*K*~*Tjeneste'* | *Tjeneste'*, *K*);
	}
	Tjeneste := *K* ~ *Tjeneste'*;

Tjeneste -> K	*K.nivå* := *Tjeneste.nivå*; > *Tjeneste* := *K*;

Tabell 1: *Et oversettelsesskjema for oversetting av strenger i spesifikasjonsspråket til et distribuert Bayesnettverk organisert i feillokaliserings- og omfangsbestemmelsesagenter.*

Oversettelsesskjemaet skal tolkes som oversettelsesskjemaet fra *kapittel 2*, men med følgende tillegg:

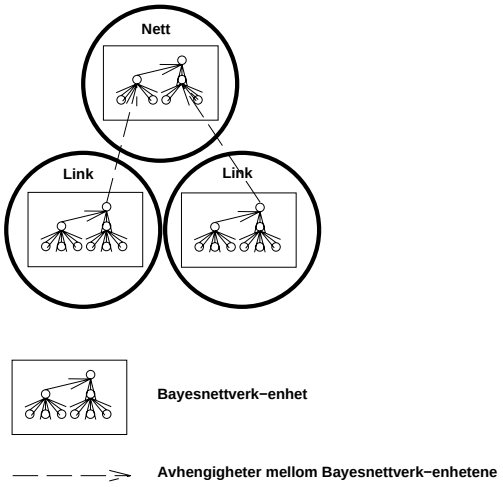
1. Hver node i avledningstreet har en ekstra attributt, *nivå*, som brukes til å holde rede på abstraksjonsnivået i spesifikasjonen av sammensetningen til en tjeneste.
2. Semantiske regler før tegnet '>' skal utføres prefiks. Semantiske regler etter '>' skal utføres postfiks.
3. Det innføres to nye typer variable i Bayesnettverket, deteksjons- og korrigeringsvariable. Disse variablene benyttes til å koordinere samarbeidet mellom feildeteksjonsagentene, feilkorrigeringsagentene og feillokaliserings- og omfangsbestemmelsesagentene. Vi tar for oss dette samarbeidet i *del 6*.
4. Det innføres en ny prosedyre, 'OppdaterEkvivalensklasser'. Denne prosedyren benytter ekvivalensrelasjonen *FKS* (beskrevet under *Identifisering av subkommunikasjonssystemer fra spesifikasjon*) til å administrere ekvivalensklassene på underliggende abstraksjonsnivå.
5. Det innføres en ny prosedyre, 'OppdaterFeillokaliserings-/omfangsbestemmelsesagenter()'. Når oversettelsen er fullført sørger prosedyren for at hvert identifisert subkommunikasjonssystem (ekvivalensklasse) har en feillokaliserings- og omfangsbestemmelsesagent. Deretter overføres strengene som hører til hver ekvivalensklasse til respektiv utplassert feillokaliserings- og omfangsbestemmelsesagent.
6. Det innføres en ny prosedyre, 'OppdaterFeildeteksjonsagent()'. Denne prosedyren sørger for at hver identifisert komponent er tildelt en fungerende feildeteksjonsagent.
7. Det innføres en ny prosedyre, 'OppdaterFeilkorrigeringsagent()'. Denne prosedyren sørger for at hver identifisert komponent er tildelt en fungerende feilkorrigeringsagent.
8. Det innføres en ny prosedyre, 'FjernOverflødigeAgenter()'. Denne prosedyren fjerne overflødige agenter som ikke lenger hører til en komponent eller et subkommunikasjonssystem.

Vi har nå definert oversettelsesskjemaet i feillokaliserings- og omfangsbestemmelsesagentene. Med dette skjemaet kan foreslått agentkontrollhierarki bygges opp automatisk og adaptivt med hensyn på den strukturelle sammensetningen til underliggende kommunikasjonssystem. Under neste punkt beskrives hvordan beregninger i resulterende Bayesnettverk-moduler kan samordnes.

5.2 Distribuerte beregninger.

Her viser vi hvordan beregningene i Bayesnettverk-modulene kan samordnes. Kort beskrevet skal beregningene i en Bayesnettverk-modul utføres uavhengig av beregningene i andre moduler, men betinget på tilstanden til *nabovariable i de andre modulene*. Med nabovariable i andre moduler menes det minimale sett av variable i andre Bayesnettverk-moduler som ved instansiering gjør sannsynlighetsfordelingen innenfor gitt Bayesnettverkmodul uavhengig av tilstanden til andre variable i det globale Bayesnettverket.

Beregningene i hver Bayesnettverk-modul kan på denne måten utføres som i en blokk i en Gibbs-sampler (se *del 5, kapittel 2* for en beskrivelse av denne beregningsmetoden). Vi har allerede foreslått en modifisert Gibbs-sampler i *kapittel 2* som ga gode feillokaliserings- og omfangsbestemmelsesresultat både ved utførte eksperiment og ved praktisk bruk. *Figur 7* illustrerer hvordan feillokaliserings- og omfangsbestemmelsesagentene på nettnivå og linknivå i *figur 5* samordner beregningene på denne måten.



Figur 7: *Distribuerte beregninger i Bayesnettverk-moduler.*

I figuren må nettagenten betinge beregningene i Bayesnettverk-modulen på variabeltilstanden til nabovariable i hver av linkagentene og omvendt. Dette for at beregningene i Bayesnettverket skal bli korrekte. Linkagentene behøver ikke å samarbeide fordi de respektive Bayesnettverk-modulene er uavhengige av hverandre gitt tilstanden til nabovariablene i nettagenten.

Beskrevet samordning kan litt forenklet organiseres ved at hver agent lagrer tilstanden til nabovariable i en buffer. Da trenger tilstander overføres kun ved tilstandsendringer.

5.3 Oppsummering.

Vi har vist hvordan et kontrollhierarki basert på et distribuert Bayesnettverk kan genereres automatisk fra spesifikasjonen av sammensetningen til tjenester. Videre er det vist hvordan beregningene i det distribuerte Bayesnettverket kan utføres.

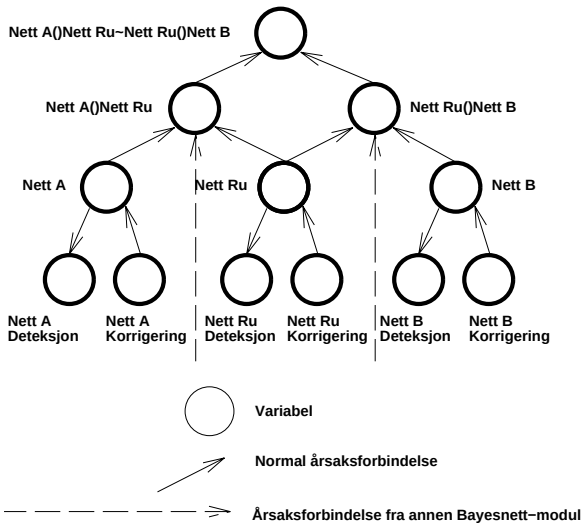
I neste del beskriver vi hvordan de ulike agentene kan operere og interaktere slik at vi oppnår global feilhåndtering.

6 Operasjon og interaksjon.

I denne delen foreslår vi hvordan feildeteksjonsagentene, det distribuerte Bayesnettverket og feilkorrigeringsagentene kan operere og interaktere. Først beskriver vi modifikasjonene på Bayesnettverket innført i oversettelsesskjemaet. Deretter beskrives operasjonen og interaksjonen mellom det modifiserte Bayesnettverket, feildeteksjonsagentene og feilkorrigeringsagentene.

6.1 Modifikasjoner i Bayesnettverket.

Det distribuerte Bayesnettverket bindes til feildeteksjonsagentene og feilkorrigeringsagentene via feildeteksjonsvariable og feilkorrigeringsvariable som vist i *figur 8*.



Figur 8: *Interaksjon mellom feildeteksjonsagenter, et distribuerte Bayesnettverk og feilkorrigeringsagenter.*

Vi tar nå for oss hver av de nye variabeltypene i figuren.

En feildeteksjonsvariabel representerer resultatet fra en aktivert feildeteksjonsagent. Årsaksammenhengene mellom tilstanden til hver komponentvariabel og tilhørende feildeteksjonsvariabel skal representere samsvaret mellom faktisk komponenttilstand og tilstanden estimert av feildeteksjonsagenten. På denne måten modelleres at en feildeteksjonsagent ikke nødvendigvis bestemmer korrekt komponenttilstand.

En feilkorrigeringsvariabel angir om respektiv feilkorrigeringsagent er aktivert eller ikke. Årsaksammenhengen mellom tilstanden til en komponentvariabel og tilhørende feilkorrigeringsvariabel representerer hvordan en aktivert feilkorrigeringsagent influerer tilhørende komponent.

Ved hjelp av disse variablene kan man oppnå interaktiv og integrert samordning mellom feildeteksjonsagentene, det distribuerte Bayesnettverket og feilkorrigeringsagentene. Vi ser på denne samordningen i neste avsnitt.

6.2 Operasjon og interaksjon mellom feildeteksjonsagenter, Bayesnettverk og feilkorrigeringsagenter.

Vi foreslår at feilhåndteringsystemet virker i en sykel på fire trinn.

1. I første trinn utføres oppdateringsprosessen beskrevet i *del 5, adaptiv oppbygging av kontrollhierarkiet*, i hver feillokaliserings- og omfangsbestemmelsesagent.
2. I andre trinn aktiverer feillokaliserings- og omfangsbestemmelsesagentene utplasserte feildeteksjonsagenter. Feildeteksjonsagentene bestemmer tilstanden til tilknyttede komponenter ved hjelp av beslutningsmodulen. Denne tilstanden instansieres så i Bayesnettverk-modulen til korresponderende feillokaliserings- og omgangsbestemmelsesagent.
3. I tredje trinn beregner hver feillokaliserings- og omfangsbestemmelsesagent de mest sannsynlige tilstandene i Bayesnettverket. Beregningene er da betinget på tilstanden til nabovariable i Bayesnettverk-modulen til foreldre- og barneagentene. På denne måten oppnås et globalt bilde av tilstanden i kommunikasjonssystemet.
4. I fjerde trinn aktiveres feilkorrigeringsagentene til komponenter hvis komponentvariable i Bayesnettverket har høy sannsynlighet for feil. Aktiverte feilkorrigeringsagenter

forsøker så å håndtere den mulige feilen. I vårt tilfellet innebærer dette en omstart eller enkel omkonfigurering. Til slutt registreres de aktiverte feilkorrigeringsagentene i Bayesnettverket slik at man ved neste sykel eventuelt kan identifisere alternative feillokasjoner hvis feil ikke er løst. Dette fordi man kan anta at en aktivert feilkorrigeringsagent vil løse eksisterende feil med en viss sannsynlighet hvis lokaliseringen er korrekt.

Deretter start en ny sykel i feilhåndteringssystemet.

6.3 Oppsummering.

Vi har nå foreslått hvordan feildeteksjonsagentene og feilkorrigeringsagentene kan samordnes med det distribuerte Bayesnettverket og dermed med feillokaliserings- og omfangsbestemmelsesagentene. I neste del tar vi for oss svakhetene og usikkerhetsmomentene ved resulterende feilhåndteringssystem.

7 Konklusjon.

Vi har til nå tatt for oss et forenklet bilde av feilhåndtering. Dette for å danne et utgangspunkt for automatisering. I denne delen ser vi på grunnleggende utfordringer som har gjort arbeidet med automatisering av feilhåndtering i kommunikasjonssystemer spennende og utfordrende.

Som vi så i *kapittel 2* er det god grunn til å tro at omfanget og lokasjonen til feil kan lokaliseres fra tilstanden til tjenester i kommunikasjonssystemer hvor sammensetningen til tjenester er statisk bestemt. Det er derimot et langt skritt fra feillokalisering og omfangsbestemmelse med denne metoden til feilkorrigering.

For det første kan ikke en feil alltid korrigeres i det punktet feillokaliserings- og omfangsbestemmelsesmetoden peker ut. For å illustrere dette beskriver vi her en feilsituasjon fra Telenors TCP/IP-SNA gatewaysystemer hvor feillokaliserings- og omfangsbestemmelsesmetoden var til hjelp, men hvor en direkte utvidelse med automatisk feilkorrigering slik vi foreslår ikke ville være hensiktsmessig. Feillokaliseringssystemet fra *kapittel 2* lokaliserte en feil til en ring i SNA-nettverket. De menneskelige systemekspertene undersøkte ringen for å finne en korrigerende handling. Det viste seg at en av de tilknyttede gatewayene blokkerte ringen. En omstart av gatewayen korrigerte feilen. Feillokaliseringssystemet ga altså en oppsummering av feilomfanget og en grov lokalisering, men denne informasjonen ville ikke være tilstrekkelig til å korrigere feilen ved vårt forslag.

Videre kan korrekt lokaliserte feil være vanskelige å korrigere. Dette fordi svært mange komponenter kan være innvolvert på innfløkte måter; flere komponenter kan for eksempel kun i gitte samspill skape feil. Å korrigere slike feil automatisk er langt utenfor det vi har foreslått

Fremgangsmåten vi foreslår er med andre ord ikke en erstatning for systemeksperter eller systemadministratorer. Det er videre et stort forbedringspotensiale.

Vi mener derimot at det er grunn til å tro at følgende er innen rekkevidde ut fra de erfaringer vi har gjort i Telenors TCP/IP-SNA gatewaysystemer. Feil som lokaliseres korrekt og som kan korrigeres ved en enkel omstart og muligens også ved enkle predefinerte omkonfigureringer bør kunne håndteres automatisk. En slik automatisering vil etter vår erfaring være til hjelp i den daglige rutinemessige driften av et kommunikasjonssystem.

Det vi er relativt sikre på er at feillokaliserings- og omfangsbestemmelsesmetoden som ble foreslått i *kapittel 2* kan lokalisere komponentfeil i kommunikasjonssystemer lik Telenors TCP/IP-SNA gatewaysystemer.

Videre er vi relativt sikre på at feillokaliseringsmetoden og omfangsbestemmelsesmetoden fra *kapittel 2* kan distribueres på en måte nær det vi foreslår her. Dette fordi man

kun trenger å overføre tilstander mellom Bayesnettverk-moduler ved tilstandsendringer og fordi tilstanden i et subkommunikasjonssystem på et gitt abstraksjonsnivå i stor grad bestemmes av lokale måleresultater.

Feildeteksjons- og omfangsbestemmelsessystem basert på sentraliserte "case"-basert resonnering har gitt gode feildeteksjons- og feilkorrigeringsresultat [5]. Dette sannsynliggjør også at automatisk omstart og muligens enkel omkonfigurering ved feil også er mulig.

Vi har nå drøftet de begrensningene vi ser ved foreslått feilhåndteringssystem. Dette feilhåndteringssystemet kan best ses på som en oppsummering av de erfaringer vi har gjort ved enkel feilhåndtering i Telenors TCP/IP-SNA gatewayssystemer. Vi håper at de grunnleggende prinsippene har fanget leserens interesse, og at verdien av videre arbeid med disse kommer frem.

8 Referanser.

[1] Fergal Somers, HYBRID: Unifiying Centralised and Distributed Network Management using Intelligent Agents, 1996 IEEE Network Operations and Management Symposium, s. 34-43.
[2] Mohsen Kahani og H.W. Peter Beadle, Decentralised Approaches for Network Management, Computer Communication Review, Volum 27, Number 3, July 1997, s. 36-47.
[3] Lundy Lewis, AI and Intelligent Networks in the 1990s and into the 21st Century, Worldwide Intelligent Systems, Approaches to Telecommunication and Network Management, 1995.
[4] B. D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, 1996, s. 265-268.
[5] Sue Abu-Hakima, The DPN Expert Advisor: Canadian AI Success Story #3, Canadian Artificial Intelligence, 1994, nr 33, s. 14-19.
[7] Stuart Russel, Artificial intelligence: a modern approach, Prentice Hall, 1995.

Kapittel 5: Konklusjon.

Ole-Christoffer Granmo

Institutt for Informatikk

Universitetet i Oslo.

10/2/99

1 Konklusjon.

Arbeidet med hovedoppgaven har spent over temaer som kommunikasjonssystemer, distribuerte samvirkende systemer samt kunskapsbaserte systemer og da spesielt Bayesnettverk. Vi avslutter hovedoppgaven med å fremheve erfaringene vi har gjort ved bruk av Bayesnettverk i kommunikasjonssystemer.

I et kommunikasjonssystem har man ofte tilgjengelig svært mange informasjonsfragmenter knyttet til prosessen bak tjenestene som tilbys. Denne prosessen kan modifiseres ved å konfigurere kommunikasjonsprogramvare, kommunikasjonshardvare og applikasjonener i kommunikasjonssystemet. En utfordring er å finne nær optimale konfigurasjoner for gitte situasjoner.

Det ideelle domenet for Bayesnettverk har følgende karakteristikker [1]:

1. Det eksisterer veldefinert kunnskap om årsakssammenhengene i domenet.
2. Årsaksammenhengene danner kjeder (dvs at en årsak har en virkning som igjen har en annen virkning, etc).
3. Hendelsene i domenet er veldefinerte.
4. Oppgaven er terapeutisk (dvs. ved å observere symptomer og risk-faktorer kan man estimere sansynligheten for ulike årsaker, og fra dette kan man avgjøre en behandling).
5. Settet av alle mulige årsaker danner en lukket verden av håndterlig størrelse.

Gitt karakteristikk *1*, *2*, *3* og *4* ser det ut til at Bayesnettverk egner seg ypperlig til feilhåndtering eller konfigurasjon av kommunikasjonsprogramvare, kommunikasjonshardvare og applikasjoner. Denne forlokkende egnetheten har i stor grad vært drivkraften bak arbeidet. På enkelte stadier har virkeliggjørelsen av feillokaliserings- og omfangsbestemmelsespotensialet ved bruk av Bayesnettverk skapt stor entusiasme og pågangsmot.

Kravet om håndterlig størrelse (karakteristikk *5*) har derimot vært kilden til frustrasjon. På stadier hvor det har vært nødvendig å utvide kompleksiteten og størrelsen til Bayesnettverkene har entusiasme og pågangsmot brått gått over i dyp frustrasjon når beregningsmetode etter beregningsmetode har feilet og hele prosjektet har stått i fare for å mislykkes.

Denne spenningen har gjort arbeidet med hovedoppgaven til en omfattende og lærerik utfordring.

Da arbeidet med beregningsmetoder til slutt førte frem også i de mer komplekse og omfattende situasjonene vi håndterte, mener vi det vil være av stor verdi å utforske mulighetene for intelligente applikasjoner og intelligent kommunikasjonsprogramvare basert på Bayesnettverk og eventuelt beslutningsnettverk [2].

Et startpunkt for en slik undersøkelse kan være følgende prosjekter:

- The Research Unit for Decision Support and Adaptation, Universitetet i Ålborg [3] samarbeider med Danish National Centre for IT Research [4] og Hewlett-Packard [5] om et automatisk beslutningssystem for kundestøtte ved Hewlet-Packard basert på Bayesnettverk. I første omgang tar dette prosjektet for seg printer-systemer som involverer HP printere og PCer med Windows 95/NT.
- Gruppen for Advanced Interactivity and Intelligence i Microsoft Research [6] driver med forskning innenfor beslutningsteori, adaptive systemer og bruk av sannsynlighetsteori og beslutningsteori til å forbedre datamaskinapplikasjoner og plattformer.
- I Bayes Net Project [7] drives blant annet forskning innenfor bruk av Bayesnettverk som beslutningsmetode for intelligente agenter.

Til slutt takker vi for leserens oppmerksomhet. Det håpes at hovedoppgaven har fanget leserens interesse og da spesielt for bruk av Bayesnettverk i kommunikasjonssystemsammenheng.

2 Referanser.

[1] Uffe Kjærulff og Finn V. Jensen, Bayesian networks, http://www.cs.auc.dk/research/DSS/profile.html
[2] Influence diagrams or Decision networks, http://www.cs.auc.dk/research/DSS/profile.html
[3] The Research Unit for Decision Support and Adaptation, Universitetet i Aalborg, http://www.cs.auc.dk/research/DSS/profile.html
[4] Danish National Centre for IT Research. http://www2.cit.dk/home.asp
[5] Hewlet-Packard, http://www.hp.com/
[6] Microsoft Research Areas, http://research.microsoft.com/scripts/main/research.asp
[7] Bayes Net Project , http://www.cs.orst.edu/~dambrosi/bayesian/frame.html