Universitetet i Oslo
Institutt for informatikk

Rapport - Hovedfag

Nettverkskartlegging i "Resilient Packet Ring"

Petter Teigen
(petterte@ifi.uio.no)

Studieretning
kommunikasjonssystemer

25. april 2003

Innhold

1 Innledning **1**
1.1 Bakgrunn . . . 1
1.1.1 Hva er RPR? . . . 1
1.1.2 Topologi . . . 2
1.2 Problemstilling . . . 2
1.3 Mål med oppgaven . . . 3
1.4 Motivasjon . . . 3
1.5 Metode . . . 3
1.6 Dokumentets oppbygning . . . 3

2 Innføring i nettverkstopologi **5**
2.1 Fysisk og logisk topologi . . . 5
2.2 Ulike topologier . . . 5
2.2.1 Maske . . . 5
2.2.2 Stjerne . . . 6
2.2.3 Buss . . . 6
2.2.4 Ring . . . 8
2.2.4.1 Innsetting, videresending, fjerning . . . 8
2.2.4.2 Kringkasting, multicast . . . 9

3 Nettverkskartlegging **10**
3.1 Hvorfor nettverkskartlegging? . . . 10
3.1.1 Nettverkskartlegging versus ruting . . . 10
3.2 Topologibildet . . . 11
3.2.1 Bygging av topologibildet . . . 11
3.2.1.1 Topologibildet som et tre . . . 11
3.2.1.2 Topologibildet som en tabell . . . 12
3.3 Seriell Buss - IEEE 1394 . . . 14

4 Innføring i RPR **15**
4.1 Generelt om de fremlagte forslagene . . . 15
4.2 Linker og rundetid . . . 15
4.3 Node . . . 15
4.4 Datatrafikk på ringen . . . 16
4.5 Samtidig bruk av ringen, “spatial reuse” . . . 16
4.6 Feilhåndtering / beskyttelse . . . 16
4.6.1 Wrapping . . . 17
4.6.2 Styring . . . 19
4.6.3 Omstokking av pakker . . . 19
4.7 Sammenkobling av RPR-ringer . . . 21
4.8 Pakkeformat . . . 21

5 Nettverkskartlegging i RPR **23**
5.1 Aspekter ved nettverkskartlegging i RPR . . . 23
5.1.1 Eget buffer for kontrollpakker . . . 23
5.1.2 Grenser til andre mekanismer i RPR . . . 24
5.1.2.1 Beskyttelse . . . 24
5.1.2.2 Valg av ring . . . 24
5.1.3 TTL i RPR . . . 25
5.2 Presentasjon av de fremlagte forslagene . . . 25
5.2.1 Darwin . . . 26
5.2.2 Utkastet . . . 27
5.3 Problemer i de fremlagte forslagene . . . 27
5.3.1 Sjekk av topologibildet . . . 27
5.3.2 Fjernet node fra ringen . . . 28
5.3.3 Koordinert nettverkskartlegging . . . 28
5.3.4 Spredning i tid . . . 28
5.3.5 Darwin . . . 28
5.3.5.1 Konsistent topologibilde . . . 30
5.3.6 Utkastene . . . 30
5.3.6.1 Fjerning av en node . . . 30
5.3.6.2 Sirkulering av pakker på ringen . . . 31
5.3.6.3 Utløsing av pakkesending . . . 32

6 Simulering og drøfting av de fremlagte forslagene **33**
6.1 Innledning . . . 33
6.1.1 Simulatorteori . . . 33
6.1.2 Bruk av simulatoren . . . 34
6.1.3 Tester . . . 34
6.1.3.1 Oppstart - test 1 . . . 35
6.1.3.2 Fjernet node ved oppstart - test 2 . . . 35
6.1.3.3 Fjerning av node - test 3 . . . 36
6.1.3.4 Lagt til node - test 4 . . . 36
6.1.3.5 Endring av identifikator - test 5 . . . 36
6.1.3.6 Bryting av en link - test 6 . . . 36
6.1.3.7 Bryting av to linker - test 7 . . . 36
6.1.3.8 Sammensetting av brutt nett - test 8 . . . 36
6.1.4 Nødvendige endringer i simulatormodellen . . . 37
6.1.5 Oppsett av simulatormodellen . . . 37
6.1.6 Usikkerheter i simulatormodellene . . . 37
6.2 Darwin . . . 38
6.2.1 Hvordan mekanismen er antatt å fungere . . . 38
6.2.2 Gjennomføring . . . 38
6.2.2.1 Oppbygging av topologibildet . . . 38
6.2.3 Resultat . . . 39
6.2.4 Drøfting . . . 39
6.2.4.1 Oppstart - test 1 . . . 39

6.2.4.2 Fjernet node ved oppstart - test 2 . . . 40
6.2.4.3 Fjerning av node - test 3 . . . 41
6.2.4.4 Fortsettelse test 3, lagt til node - test 4 og endring av identifikator - test 5 41
6.2.4.5 Bryting av en link - test 6 . . . 41
6.2.4.6 Bryting av to linker - test 7 . . . 43
6.2.4.7 Sammensetting av brutt nett - test 8 . . . 43
6.3 Utkast 0.3 / 1.0 . . . 43
6.3.1 Hvordan mekanismen er antatt å fungere . . . 43
6.3.2 Gjennomføring . . . 44
6.3.3 Problemer og mangler under implementering . . . 45
6.3.3.1 Wrapping . . . 45
6.3.3.2 Konsistenssjekk av topologibilde . . . 45
6.3.4 Resultat . . . 46
6.3.5 Drøfting . . . 46
6.3.5.1 Oppstart - test 1 . . . 46
6.3.5.2 Fjernet node ved oppstart - test 2 . . . 46
6.3.5.3 Fjerning av node - test 3 . . . 48
6.3.5.4 Lagt til node - test 4 . . . 48
6.3.5.5 Endring av identifikator - test 5 . . . 48
6.3.5.6 Bryting av en link - test 6 . . . 49
6.3.5.7 Bryting av to linker - test 7 . . . 51
6.3.5.8 Sammensetting av brutt nett - test 8 . . . 51

7 Presentasjon av egen metode for nettverkskartlegging 53
7.1 Diskusjon . . . 53
7.2 Designkriterier . . . 53
7.3 Beskrivelse av mekanismen for nettverkskartlegging . . . 54
7.3.1 Statusvariabel . . . 55
7.3.2 Oppstart . . . 55
7.3.3 Mottak av topologipakke på samme ring som den ble sendt . . . 55
7.3.4 Mottak av topologipakke på motsatt ring enn der den ble sendt . . . 55
7.3.5 Oppførsel ved brutt link . . . 56
7.3.6 Sending av topologipakker . . . 56
7.3.7 Topologi timer . . . 58

8 Simulering og drøfting av egen mekanisme 59
8.1 Hvordan mekanismen er antatt å fungere . . . 59
8.1.1 Ved oppstart . . . 59
8.1.2 Fjerning av en node, ingen wrapping . . . 59
8.1.3 Endring av identifikator på en node . . . 60
8.1.4 Bryting av link, wrapping . . . 60
8.1.5 Sammensetting av brutt, wrappet link . . . 60
8.2 Testkjøringer under implementering . . . 60
8.2.1 Tapt topologipakke . . . 60
8.2.2 Endring av identifikator på en node . . . 60

8.2.3 Konsistenssjekk . . . 61
8.2.4 Fjerning av en node, ingen wrapping . . . 62
8.2.5 Bryting av linker, wrapping . . . 62
8.2.6 Sirkulerende topologipakker på ringen . . . 63
8.3 Resultat . . . 63
8.4 Drøfting . . . 64
8.4.1 Oppstart - test 1 . . . 64
8.4.2 Fjernet node ved oppstart - test 2 . . . 64
8.4.3 Fjerning av node - test 3 . . . 65
8.4.4 Lagt til node - test 4, og endring av identifikator - test 5 . . . 67
8.4.5 Bryting av en link - test 6, og bryting av to linker - test 7 . . . 68
8.4.6 Sammensetting av brutt nett - test 8 . . . 68
8.5 Har mekanismen oppfylt designkriteriene? . . . 71

9 Sammenligning av mekanismene for nettverkskartlegging 73
9.1 Testresultater . . . 73
9.1.1 Testene . . . 73
9.1.2 Generelt . . . 75
9.2 Nettverkskartlegging i RPR versus IEEE 1394 . . . 76

10 Konklusjon og videre arbeid 77
10.1 Generelt . . . 77
10.2 Er problemstillingene løst? . . . 77
10.3 Er oppgavens mål løst? . . . 78
10.4 Videre arbeid . . . 78

11 Ordbok 80

Referanser 81

APPENDIX 83

A Figur 6.11 i kapittel 6.5.2 i utkast 0.3 83

1 Innledning

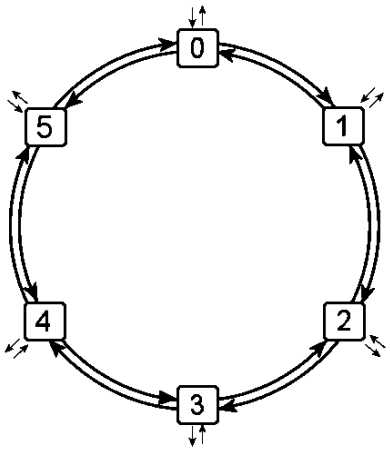
1.1 Bakgrunn

På grunn av Internetts enorme størrelse må nettet deles opp i mindre deler. Disse delene kan man kalle Internetts byggestener. De danner Internetts fundament slik at datamaskiner på motsatt side av jorda kan nå hverandre. Internett er satt sammen av mange ulike slike byggestener som da blir en analogi på forskjellige nettverkstyper som for eksempel Ethernet, Token Ring og ATM. Disse nettverkstypene er satt sammen av andre byggestener som rutere, svitsjer og datamaskiner til brukere. Etter hvert som nettverksapplikasjoner og maskinvare utvikler seg vil applikasjonene, maskinvaren og brukerne av systemene kreve mer av nettverket. På grunn av dette må også Internetts byggestener være i utvikling. En slik byggesten er RPR.

1.1.1 Hva er RPR?

Resilient Packet Ring (RPR) er et standardiseringsprosjekt i regi av IEEE (Institute of Electrical and Electronics Engineers). RPR er en ny nettverksteknologi i 802 LAN/MAN serien og har fått betegnelsen 802.17. I den samme serien finnes blant annet kjente teknologier som Ethernet (802.3), Token Ring (802.5) og trådløst lokalnett (802.11). Ethernet er i dag veldig populært og brukes i alt fra små hjemmenettverk til store bedriftsnettverk. Blant pådriverne i standardiseringsprosjektet finner vi blant annet Cisco Systems. Cisco er velkjent for sine ulike produkter innen datakommunikasjon. Selve prosjektet er et samarbeid mellom flere bedrifter og organisasjoner som deltar i forskjellig grad.

Tanken bak RPR er å forbedre de nettverkstypene som finnes i dag. RPR skal, ved å bruke eksisterende fysisk lag, oppnå maksimal utnyttelse av de ressursene som finnes på nettet [11].



Figur 1: En enkel RPR-ring med to unidireksjonale ringer.

Figur 1 viser en enkel RPR-ring. Datamaskiner på ringen, fra nå kalt noder, kobles sammen til å danne en ringstruktur. Det skal i utgangspunktet være to ringer som danner en RPR-ring. Datatrafikken går på begge ringene, men i motsatt retning.

RPR vil bli grundigere gjennomgått i kapittel 4 på side 15.

1.1.2 Topologi

Datanettverk er dynamiske. Noder kommer til og forlater nettverkene. I noen nettverkstyper kan det være viktig for nodene i nettverket å vite om de andre nodene i nettet. Det finnes flere grunner til dette, og noen av de er tatt opp i kapittel 3 på side 10. Koblingene og rekkefølgene av nodene i et nettverk kalles nettets topologi. I en RPR-ring har hver node et topologibilde som gjenspeiler ringens topologi.

Nodene i RPR og andre nettverk må ha en mekanisme for å finne ut nettets topologi. Denne mekanismen kalles på engelsk "topology discovery", og kan oversettes til norsk som nettverkskartlegging. Det er viktig at nettverksteknologien tilbyr en robust, fleksibel, enkel og stabil måte å gjennomføre nettverkskartlegging på. Det er nødvendigvis ikke like lett å oppfylle alle disse kravene på en gang. En enkel metode for nettverkskartlegging behøver for eksempel ikke være robust. Nettverkskartlegging må til enhver tid sørge for et feilfritt topologibilde i alle noder uten å legge for stor last på nettet. Utifra hvilken måte man velger å løse problemet på kan det gi forskjellig last på nettet og yte annerledes avhengig av hvilke krav man setter.

1.2 Problemstilling

Problemstillingene i denne oppgaven handler om nettverkskartlegging på RPR-ringen. Oppgaven går ut på å finne en god løsning på problemet. Aktuelle problemstillinger innen nettverkskartlegging på en RPR-ring er:

- Hvordan vet en node i en RPR-ring hvilke andre noder som finnes på ringen?
- Hvordan vet nodene i en RPR-ring at det har blitt satt inn en ny node i ringen?
- Hvordan vet nodene i en RPR-ring at det har blitt fjernet en node fra ringen?
- Hvordan vet nodene i en RPR-ring at en node på ringen har feilet?
- Hvordan vet nodene på ringen at en link mellom to noder har feilet?
- Om alle nodene i ringen er slått av, for deretter å bli slått på. Hvordan startes kommunikasjon mellom noder på ringen?

I starten av hovedfagsperioden var RPR fortsatt på standardiseringsbordet. Det var ikke fastsatt en metode for nettverkskartlegging i RPR, men det kom flere forslag på løsning av problemet. Disse forslagene ble lagt frem av forskjellige grupperinger og bedrifter i RPR standardiseringsgruppen. Som en del av denne forskningsoppgaven skal noen av de fremlagte forslagene vurderes separat og opp mot hverandre for å finne styrker og svakheter. De ulike måtene å løse problemet på kan ha ulike egenskaper utfra hvilke evalueringskriterier man setter. Flere problemstillinger for hovedfagsarbeidet dukker da opp:

- Hva er styrkene og svakhetene til de presenterte forslagene, og hva gjør de gode eller mindre gode?
- Hvilke likheter og forskjeller finnes i forslagene?
- Hva argumenterer og vektlegger forslagsstillerne for i sine forslag, og hvordan?

1.3 Mål med oppgaven

Det er flere mål med å gjennomføre denne oppgaven. Det mest åpenbare målet er å få innsikt i hvordan man gjennomfører nettverkskartlegging på RPR-ringen. Kjennskap til selve RPR-teknologien er da en nødvendighet for å gjennomføre oppgaven. Rapporten tar for seg presentasjon, evaluering og sammenligning av ulike forslag for nettverkskartlegging og vil derfor gi leseren mulighet til å se på forslagene som er lagt frem med en bredere kunnskap om mekanismen.

1.4 Motivasjon

Oppgaven setter lys på flere sider ved nettverksteknologi. Standardiseringsprosessen gir innsikt i alle deler av spesifiseringen av en ny teknologi. Selv om en slik prosess gjennomgår alle deler, behøver man ikke å sette seg grundig inn i alle for å forstå hva RPR handler om. Denne oppgaven fokuserer på nettverkskartlegging og de mekanismene som er direkte knyttet til dette. Jeg synes nettverk og forskjellige topologier er et interessant tema, og da kunne det passe bra å fordype seg i en spesiell nettverkstype. I starten av hovedfagsperioden var det tid til å kunne påvirke arbeidsgruppen hvis det kom opp noe som de kunne ha nytte av i arbeidet. Muligheten til å påvirke standardiseringsarbeidet var en pådriver for arbeidet med oppgaven.

1.5 Metode

Det er flere måter å gå løs på problemstillingene på. Denne oppgaven vil dreie seg mye om simuleringer av RPR-ringer der mekanismene for nettverkskartlegging blir testet. Simuleringer er valgt fordi det gir en modell av systemet som er enkel å endre og tilpasse. Siden RPR foreløpig er på standardiseringsbordet finnes det ingen maskinvare for RPR som de ulike mekanismene for nettverkskartlegging kan testes på. En annen mulighet hadde vært å analysere hvordan mekanismene fungerer ved å ta i bruk matematiske bevis. Dette er ikke gjort på grunn av mekanismene og RPR-ringen er for komplisert.

1.6 Dokumentets oppbygning

Rapporten begynner med innføring i diverse bakgrunnskunnskap om nettverk, RPR og nettverkskartlegging. Fra å begynne med generell kunnskap om forskjellige nettverkstopologier går vi videre og snakker om nettverkskartlegging. Deretter vil RPR og nettverkskartlegging i RPR diskuteres før presentasjon og simuleringer av de forskjellige metodene og drøfting av resultatene fra disse testene bygger opp til en konklusjon. De fremlagte forslagene vil presenteres og simuleres før min egen metode for nettverkskartlegging blir omtalt. Dette kommer av at resultater fra simuleringer av de fremlagte forslagene er grunnlag for det som står i kapitlene om min egen metode. På slutten av rapporten er det presentert en konklusjon som inkluderer blant annet oppsummering og videre arbeid.

Hvert kapittel vil gi en liten innsikt i hva som følger. På slutten av oppgaven finnes det en ordbok. Denne ordboken inneholder utvalgte ord og uttrykk som brukes i oppgaven. Oppgaven tilstreber å bruke norske ord og uttrykk. Noen betegnelser som ofte benyttes på engelsk i norsk tale er oversatt til norsk i denne oppgaven. Dette er nevnt i de enkelte tilfellene.

Bakerst i rapporten er det lagt ved en CD. Denne CD'en inneholder all Java-koden som er brukt under simuleringene tilknyttet denne oppgaven. Det er også en "README"-fil for hver katalog som forklarer litt om koden og hvordan den eventuelt kan kjøres. CD'en inneholder også denne rapporten i pdf-format.

2 Innføring i nettverkstopologi

Dette kapittelet presenterer ulike topologier med hjelp av illustrasjoner. Fordeler og ulemper med de forskjellige måtene å bygge opp et nett på vil også diskuteres der det er hensiktsmessig.

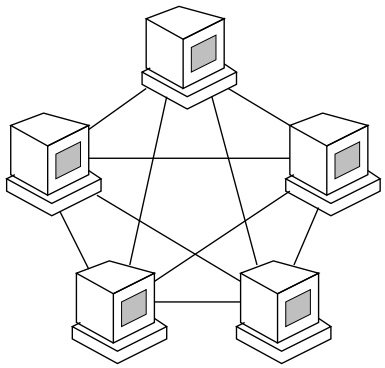
2.1 Fysisk og logisk topologi

Det er to måter å se på topologien til et nett på. Den ene er den fysiske topologien som forteller hvordan nodene i nettet er koblet sammen med ledninger. Den logiske topologien kan deles opp i flere nivåer. Den logiske topologien forteller blant annet hvordan nettet ser ut med tanke på hvordan dataene beveger seg over nettet. Hvis man ser på den logiske topologien fra et høyere lag kan man tenke seg at man ikke ser hvordan dataene beveger seg gjennom nettet, men at de kommer ut av nettet der de skal. Den logiske og den fysiske topologien trenger nødvendigvis ikke å være den samme for ett og samme nett. Et nett som har en fysisk stjernetopologi kan for eksempel oppføre seg som en buss eller ring. De ulike topologiene blir forklart under.

2.2 Ulike topologier

RPR-nettverket er bygd opp i en fysisk ringtopologi. Avhengig av hvilket lag man ser fra er RPR-ringen også en logisk ringtopologi. Fra høyere lag kan man derimot se på RPR som et nett der man ikke vet hvordan nettet er bygget opp. Det er flere andre måter å bygge opp et nettverk på. Jeg vil her presentere noen andre topologier og se på de grunnleggende forskjellene på de ulike topologiene. Et nettverk kan også bygges opp av flere av de ulike strukturene som nevnes i dette kapittelet.

2.2.1 Maske



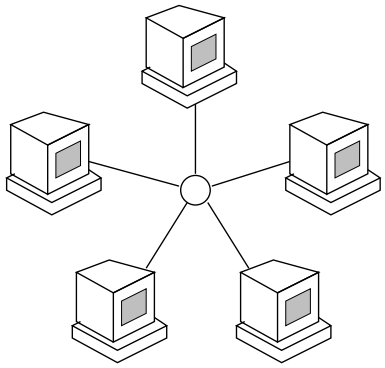
Figur 2: Et fullstending sammenkoblet maskenettverk.

Figur 2 viser et nettverk som er koblet sammen i til et maskenettverk. Det finnes to typer maskenettverk. Det ene er det fullstendig sammenkoblede og det andre er delvis sammenkoblet. I det fullstendige sammenkoblede, som på figuren, har alle nodene i nettet en kobling til

alle de andre nodene i nettet. Denne typen nettverk er effektive med tanke på sending av data fordi dataene kan hele tiden gå direkte til mottakeren. Det er også stor redundans på veier til de andre nodene i nettet hvis det skulle oppstå en feil på den korteste linken. Antall linker i et fullstending sammenkoblet maskenettverk øker eksponensielt. Hvis det er N noder i nettet må det $\frac{N(N-1)}{2}$ linker til for å koble sammen alle nodene. Dette er en ulempe fordi det kreves mye for å sette opp et slikt nett, særlig når nettets størrelse øker.

I et delvis sammenkoblet maskenettverk har noen av nodene link til alle de andre nodene i nettet. Det vil derimot være noder som er koblet som grener inn i dette nettet slik at ikke alle har link til alle de andre nodene. I et slikt nett vil det være enklere å legge til noder, da det ikke er nødvendig å trekke en link til alle de andre nodene i nettet.

2.2.2 Stjerne

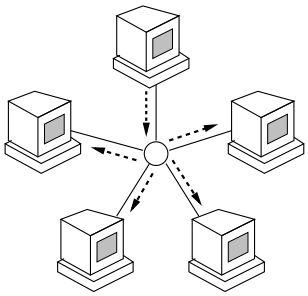


Figur 3: Et typisk stjernenettverk.

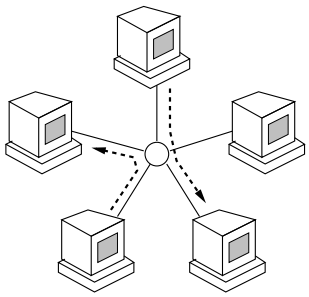
Figur 3 viser et typisk stjernenettverk. Her er alle nodene i nettet koblet til ett samlingspunkt, også kalt konsentrator. All trafikk som går på nettet må gjennom denne noden for å nå en av de andre nodene på nettet. Noden i samlingspunktet er ofte en hub eller en svitsj. En hub har den egenskapen at alt som sendes fra en node i retning hubben vil sendes videre på alle de andre linkene koblet til noden uansett mottakeradresse. Dette er illustrert i figur 4 på neste side. Ved bruk av en hub vil kun én node kunne sende av gangen. En svitsj vil sende pakker som kommer fra en node kun til den linken der mottakeradressen finnes. Hvis svitsjens design tillater det vil flere noder kunne sende til hverandre samtidig hvis det er to forskjellige avsendere og to forskjellige mottakere. Dette er vist i figur 5 på neste side. Et stjernenettverk er enkelt å sette opp men er sårbart for feil. Hvis det oppstår en feil i konsentratoren slik at denne ikke kan sende og motta pakker vil nettet være ubrukelig for alle de andre nodene. Ved høy datatrafikk på nettet vil noden i midten være en flaskehals der det kan oppstå metning fordi alle sender via det samme punktet.

2.2.3 Buss

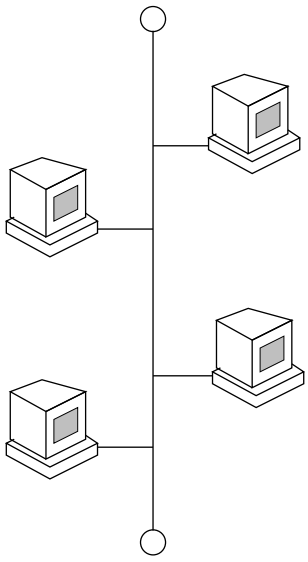
I figur 6 på neste side ser man at alle nodene er koblet til samme link. Når alle nodene er koblet til samme fysiske link på denne måten kaller vi det en buss.



Figur 4: Stjernetopologi med hub.



Figur 5: Stjernetopologi med svitsj.

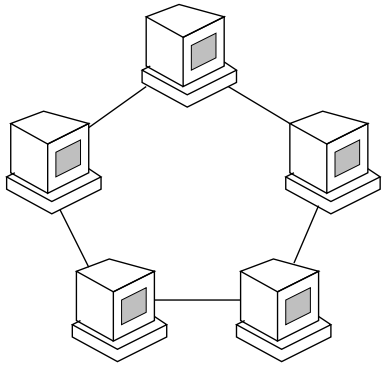


Figur 6: Alle nodene koblet til samme link på en buss.

En buss er enkel å sette opp og en grei løsning for et lite antall maskiner. Det er også lett å koble til noder med en slik topologi uten å bryte noen forbindelser mellom noen andre noder i nettet.

En felles buss fører til at bare én node kan sende data til nettet på en gang. Hvis flere sender vil pakkene kollidere på linken og skape støy. Ytelse på denne typen nett vil bli dårligere når antall noder øker [10]. En annen ulempe med slike nett er sårbarheten. Det finnes ingen linker som kan ta over dersom hovedlinken blir brutt. Hvis dette skulle skje vil nettet bli splittet og noen deler av nettet ikke få kontakt med andre deler av nettet.

2.2.4 Ring



Figur 7: Bildet viser strukturen i et ringnettverk.

Figur 7 viser et enkelt ringnettverk. Det er flere ulike typer ringnettverk. De velkjente er Token Ring (IEEE 802.5) og FDDI (Fiber Distributed Data Interface, ANSI X3T12). I starten av oppgaven så vi at RPR har to ringer der datatrafikken går hver sin vei på de to ringene. FDDI og Token Ring er også bygget opp med to ringer. Det er derimot en primær og en sekundær ring. Den sekundære kan brukes til datatrafikk, som sikkerhet hvis den primære skal feile eller kun til kontrolltrafikk på ringen. I RPR er de to ringene likeverdige der nodene velger den korteste veien til mottakeren. Det er også andre ting som skiller RPR fra de andre typene ringnettverk. I RPR er det mottakeren som tar pakkene av ringen. Dette gir mulighet for samtidig bruk av RPR-ringen, noe som gjør at flere noder kan sende samtidig. Samtidig bruk av RPR-ringen er grundigere forklart i kapittel 4.5 på side 16. I FDDI og Token Ring har bare én node mulighet til å sende av gangen. Dette kommer av at hvis nodene skal sende må de først få tak i et token, et spesielt bitmønster, som sirkulerer på ringen før de kan sende. Når de så sender går pakkene helt rundt ringen der det er avsenderen som tar pakkene av ringen. Mottakeren(e) langs ringen kopierer dataene når de passerer.

2.2.4.1 Innsetting, videresending, fjerning Transporten av datapakker over en ringstruktur er enkel. Det kreves litt prosessering når pakken skal legges på ringen. Dette for å vite hvilken retning på ringen pakken skal gå. Når først pakken er kommet på ringen vil det derimot være enkle valg som skal gjøres i nodene langs veien til mottakeren. Hodet prosesseres for å finne ut om pakken skal sendes videre langs ringen, eller om den skal tas av ringen i

den aktuelle noden. Ved andre nettverkstyper, som for eksempel maskenettverk, vil det kunne foregå ruting som bestemmer hvilken link pakken skal sendes videre på. Dette vil skje i hver node frem til mottakeren, noe som vil kunne være mer tidkrevende enn en enkel sjekk på om pakken skal til denne noden eller ikke.

2.2.4.2 Kringkasting, multicast Hvis alle eller et sett av nodene i en ringstruktur skal ha tak i en pakke som sendes fra en av nodene på ringen kalles dette henholdsvis kringkasting og multicast. På en ringstrukur er dette enkelt da avsendernoden kun trenger å sende én pakke. Når denne pakken når mottakerne langs ringen vil disse kopiere innholdet i pakken mens den kan fortsette til andre potensielle mottakere videre på ringen. Hvordan man skiller mellom unicast (pakker med én mottaker) og kringkasting av pakker blir spesifikt for hver nettverkstype. Ethernet definerer dette med en spesiell adresse i mottakerfeltet.

Oppsummering

Maske, stjerne, buss og ring er fire måter å bygge opp et nett på. Det finnes også spesialtilfeller av noen av topologiene. Topologiene har forskjellige egenskaper når det gjelder antall veier til de andre nodene, sårbarhet og hvor avanserte de er å sette opp.

3 Nettverkskartlegging

Dette kapittelet vil ta for seg nettverkskartlegging generelt. Det starter med hvorfor nettverkskartlegging er nødvendig i ulike nettverk. Deretter blir det foreslått noen metoder for å strukturere informasjonen mekanismen tilbyr. Til slutt vil det presenteres en metode som er ulik RPRs måte å løse problemet på. Nettverkskartlegging i RPR blir senere sammenlignet med denne.

3.1 Hvorfor nettverkskartlegging?

Noder i nettverk har nytte av å kjenne til nettets topologi. Denne informasjonen kan være nyttig i noen tilfeller når nodene skal sende data til hverandre. Dette avhenger av hvilken nettverkstype man har. I en ringstruktur vil dataene nå mottakeren langs begge veiene rundt ringen, men en av veiene kan være raskere enn den andre. Hvis en node A skal sende data til node B må A vite hvor i nettet B befinner seg for å sende data i riktig retning. Hvis det er flere linker data kan videresendes på i en node må eventuelle noder mellom A og B vite hvor B befinner seg for å kunne videresende data som er sendt fra A. I en ringstuktur der hver node har to utganger kan også mellomliggende noder bare videresende pakken.

Kartlegging av topologi i et nettverk vil være interessant for å være motstandsdyktig for feil. Hvis det oppstår en feil på en link vil kjennskap til nettets topologi kunne gi de andre nodene i nettet en mulighet til å sende datatrafikk utenom feilen. Dette avhenger av at nettet er bygget opp slik at det finnes en annen vei til en potensiell mottaker av data. Ringnettverk har denne muligheten. I en ring vil det typisk finnes to veier til alle nodene på ringen, forutsatt at det ikke er noen feil på ringen. I de fleste tilfeller vil derimot avstanden til nodene være forskjellig på de to ringene og den ene veien derfor foretrekkes fremfor den andre. I tilfelle en feil oppstår kan det likevel være interessant å nå noden langs den lengre veien for å komme frem til den. Dette vil være en situasjon der nodene i nettet vil ha nytte av å vite om nettets topologi slik at de kan sende data den lengre veien.

En kan også tenke seg situasjoner der det ikke nødvendigvis er hensiktsmessig for alle noder i et nett å vite om nettets topologi. Det kan være at denne informasjonen er rett og slett unødvendig eller at nettet er så stort at det blir altfor mye informasjon som blir lagret. Dette vil avhenge av både den fysiske og den logiske topologien.

I et ringnettverk som RPR kan man også vurdere om det er nødvending med en slik mekanisme. Det er klart det er flere veier å sende til samme node i et ringnettverk og at den korteste veien er mest hensiktsmessig å sende på. Men det behøver nødvendigvis ikke være viktig at noden finner den korteste veien før man kan sende pakker på ringen. Avhengig av nettets størrelse og last kan man vurdere om det kan være like hensiktsmessig å generere slik topologiinformasjon når det trengs.

3.1.1 Nettverkskartlegging versus ruting

Basert på det overstående kapittelet kan man tenke på hva som skiller nettverkskartlegging og ruting fra hverandre. Grensen mellom de to er uklar. Det er derimot noen forskjeller som kan nevnes. Nettverkskartlegging er en mekanisme for at nodene på nettet skal finne nettets topologi. Topologibildet som mekanismen bygger opp kan brukes til blant annet å bestemme hvor

pakker skal videresendes. Ruting derimot er både oppbygging av rutingtabell (topologibilde) og delen som omhandler videresending av data.

Endringer som skjer i et topologibilde vil påvirke flyten av data på nettet umiddelbart etter det er endret. Endringer i nettets topologi som ruting må ta hensyn til behøver derimot ikke å endres umiddelbart. Dette kan komme av at nettet kan være mere avansert sammenkoblet enn hvordan for eksempel RPR er sammenkoblet. Det behøver nødvendigvis ikke være nødvendig for alle nodene å oppdatere sin rutingtabell ved alle endringer i topologien. Det kan også være at delen som omhandler videresending av data ikke sjekker hvor data skal videresendes for hver pakke som kommer inn, men sjekker opp mot rutingtabellen med jevne mellomrom.

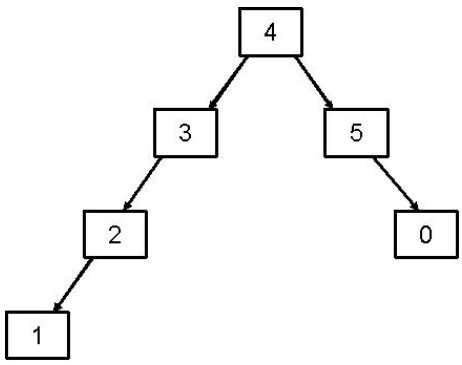
3.2 Topologibildet

Med den informasjonen nettverkskartlegging tilbyr bygger nodene opp et topologibilde. Det er viktig at alle nodene i nettverket til enhver tid har et riktig topologibilde. Hvis en node har et topologibilde som ikke stemmer med virkeligheten kan det føre til uønsket aktivitet på nettet. Dette kunne vært unngått hvis noden hadde korrekt topologiinformasjon. En node kan for eksempel ha en node i sitt topologibilde som ikke finnes i det virkelige nettet. Det kan da være fare for at noden vil sende data til en slik ikke eksisterende node.

3.2.1 Bygging av topologibildet

Topologibildet kan være oppbygd i forskjellige former. Noen metoder er foreslått og diskutert under. De foreslåtte metodene kan brukes i flere nettverkstyper. Eksempelene som er vist er derimot vist med RPR.

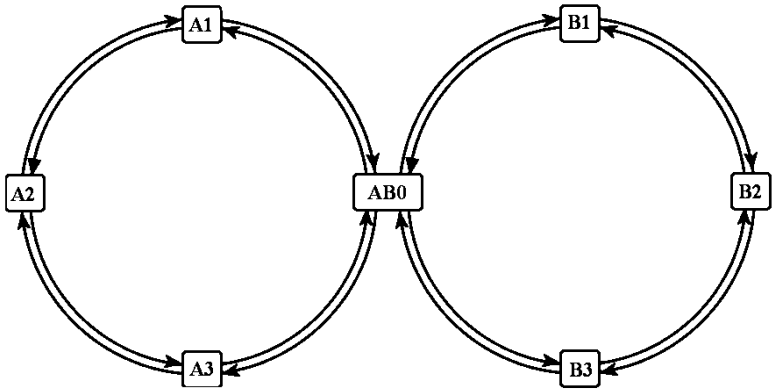
3.2.1.1 Topologibildet som et tre Topologibildet kan være i form av et tre. Rekkefølgen av nodene i treet er spesifikt for hver node. Treet inneholder alle nodene den aktuelle noden kan nå. Figur 1 på side 1 viser et bilde av en enkel RPR-ring. Hvis node 4 skal bygge opp et tre av nodene på ringen ville det kunne se ut som i figur 8. Det er flere løsninger på hvordan treet kan se ut etter oppbygging. Det som er presentert er bare et forslag på hvordan det kan se ut.



Figur 8: Her har node 4 bygget opp et tre av nodene i figur 1 på side 1.

Vi ser her at node 4 har bygget opp et tre der alle nodene på ringen er med. Treet er bygget slik at noden selv er plassert i roten av treet og alle de andre nodene er plassert nedover i treets grener. Noden har da fått et topologibilde av hvordan nettet ser ut og hvordan noden skal kunne nå de andre nodene i nettet. Hvis node 4 for eksempel skal nå node 2 vet den at denne er tilgjengelig gjennom node 3.

Vi ser av figuren at det treet noden bygger opp blir veldig enkelt. Alle nodene i treet har maksimalt en forelder og ett barn. Noden i toppen av treet, rotnoden, har kun barn. Treets enkle struktur er på grunn av RPR-ringens topologi. Problemstillingen rundt oppbygging av et tre blir ganske annerledes ved mer avansert koblede nett. I RPR kan dette være når ringer sammenkobles som i figur 9. Sammenkobling av RPR-ringer vil bli grundigere forklart senere i oppgaven.



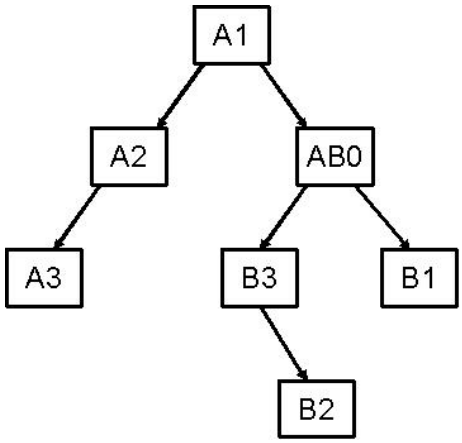
Figur 9: Her er to ringer koblet sammen.

Man kan tenke seg at nodene på alle ringene som er koblet sammen skal inn i treet for hver node. La oss si at node A1 i figur 9 skal bygge opp et tre. Den vil da kunne ta med alle nodene, også de som hører til i ring B. Treet node A1 bygger opp vil kunne se ut som i figur 10 på neste side. At nodene kjenner til topologiinformasjon om ringen de ikke er direkte koblet til er ikke støttet av RPR. En kan derimot tenke seg sammenkoblede RPR-ringer kan ha denne funksjonaliteten.

En kan også tenke seg at flere enn to ringer er koblet sammen, slik at treet etterhvert kan bli ganske avansert. Flere enn to ringer vil også kunne kobles til samme node.

3.2.1.2 Topologibildet som en tabell En annen måte å bygge opp topologibildet på er som en tabell. Utkast 1.0 [6] har presentert et eksempel på hvordan dette kan gjøres. Presentasjon av de ulike utkastene blir gjort i neste kapittel. Her bygges det opp en tabell for hver ring som kan se ut som figur 11 på neste side. Her har node nummer 4 i figur 1 på side 1 bygget opp en tabell for sin ytre ring. En tilsvarende tabell må også finnes for den indre ringen hvis topologibildet skal være komplett. De to siste kolonnene i tabellen er ikke fylt inn. De er ikke en viktige for å se prinsippet for hvordan man bygger opp topologibildet i en tabell.

En rad i topologibilde blir senere i denne oppgaven også kalt en post.



Figur 10: Her har node A1 bygget opp et tre av nodene på begge ringene.

Avstand	Lokal MAC adr.	Øst MAC adr	Vest MAC adr	Nedstrøms link tilgjengelighet	Nedstrøms link allokert klasse A Båndbredde
0	4	5	3	– – –	– – –
1	5	0	4	– – –	– – –
2	0	1	5	– – –	– – –
3	1	2	0	– – –	– – –
4	2	3	1	– – –	– – –
5	3	4	2	– – –	– – –

Figur 11: Dette er et eksempel på hvordan topologibildet kan bygges opp som en tabell.

3.3 Seriell Buss - IEEE 1394

IEEE har laget en standard for høyhastighets seriellbuss. Denne er ofte kalt “Firewire” eller bare “1394” (kommer av nummeret på standarden). Her kan noder i nettet kobles sammen vilkårlig og hver node kan ha en eller flere porter som er koblinger til andre noder. IEEE 1394 har en metode for nettverkskartlegging som er grundig forklart i standarden. Denne skiller seg fra de metodene for RPR vi kommer til å møte senere i rapporten på mange punkter. Mot slutten av denne rapporten, i kapittel 9.2 på side 76, vil metodene for nettverkskartlegging sammenlignes i IEEE 1394 og RPR.

IEEE 1394 opererer med tre typer noder i sitt nett. En node som ikke er koblet til noen kalles isolert, en løvnode har 1 oppkoblet port og grennoder som har to eller flere oppkoblede porter. Rotnoden som velges (forklart senere) vil kunne være enten en løvnode eller en grennode.

Algoritmen er delt i 4 (3+1) deler. Buss initiering, oppsett av tre (tree-identify) og selvidentifisering (self-identify) er de tre innledende fasene etterfulgt av oppbygning av topologiinformasjon om nettet. Hvis det skjer en endring i nettets topologi ved at en node legges til eller fjernes kjøres hele denne prosessen på nytt. Den første fasen sletter all lagret topologiinformasjon i nettet. Noden vil da også finne ut hva slags node den selv er (løv/gren). Den andre fasen bygger opp et tre av de sammenkoblede nodene. Løvnodene i nettet vil sette i gang en mekanisme som til slutt finner én rotnode i nettet og alle nodene vet også hvilket forhold de har til sine tilkoblede porter. En port kan ha to muligheter, den kan være forelder eller barn for sin tilkoblede node. En nodes forelder er nærmere roten enn en nodens barn. Roten vil så i den tredje fasen sette i gang en prosess der alle nodene i nettet får en identifikator. Disse vil begynne på 0 og gå oppover til roten selv får den høyeste verdien. Metoden for å finne ut de enkeltes identifikator kan man tenke på som et token som traverserer treet i en slik rekkefølge der man for alle noder, går rekursivt nedover og håndtérer først barn (i stigende rekkefølge) og så noden selv. Under del tre vil nodene, etterhvert som de får en identifikator, fortelle de andre hvilken identifikator de har fått. På denne måten kan nodene telle hvor mange som har fått tildelt sin identifikator. Når det så blir deres tur til å sette sin identifikator vil det være denne telleren økt med 1.

Oppsummering

Kapittelet har tatt for seg ulike aspekter ved nettverkskartlegging. Vi har sett at det er ønskelig med kjennskap til nettets topologi for å utveksle data på en effektiv måte og være motstandsdyktig mot feil. En trestruktur og tabell er to mulige måter å strukturere informasjonen når et topologibilde skal bygges opp.

4 Innføring i RPR

Dette kapittelet vil ta for seg RPR-ringen og noen av mekanismene på ringen. Kapittelet vil gi en grunnleggende innføring i RPR for å forstå resten av denne oppgaven før senere kapitler går grundigere inn i mekanismer for nettverkskartlegging som kan opererere på RPR-ringen.

4.1 Generelt om de fremlagte forslagene

RPR-arbeidsgruppen har lagt frem flere forslag til standarden. Disse kalles på engelsk “draft” og vil i denne oppgaven oversettes til norsk som “utkast”. Denne oppgaven vil baseres på to av disse utkastene, utkast 0.3 og utkast 1.0. De er begge to tatt med på grunn av mekanismen for nettverkskartlegging i de to. Utkast 1.0 er en forbedret versjon av 0.3 der noen feil og mangler blir rettet opp. 0.3 er tatt med for å drøfte de feil og mangler som finnes her. I flere tilfeller i denne oppgaven brukes bare “utkastet” der det siktes til det som er felles for utkast 0.3 og utkast 1.0. De to utkastene er kun tilgjengelig for medlemmer av arbeidsgruppen.

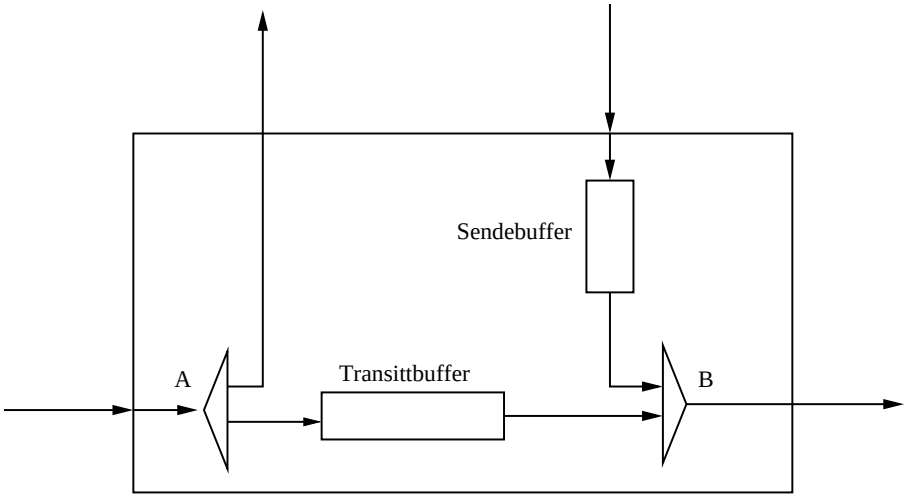
4.2 Linker og rundetid

Nodene på RPR-ringen er koblet sammen med linker. Hver link har en propagasjonstid som er tiden ett bit bruker på å komme fra den ene noden til den neste. Propagasjonstiden er proporsjonal med den fysiske avstanden mellom to noder. Forskjellige avstander gir derfor ulike tider. Den totale propagasjonstiden for alle linkene i en ring blir i denne oppgaven kalt rundetid. For å lette senere presentasjoner av simuleringsresultater er det hensiktsmessig å definere et symbol for en rundetid. Forkortelsen RTT (eng. Round Trip Time) vil bli brukt til dette.

4.3 Node

Figur 12 på neste side viser en skisse av en node. Det som vises er nodens grensesnitt for én ring. Siden en RPR-ring har to ringer er det to identiske slike i hver node, en for hver retning. I figuren kommer data inn fra venstre side. Pakkene som ankommer denne noden kommer først inn i en mekanisme som avgjør om pakken skal av ringen i denne noden (A på figuren). Hvis pakken skal av, sender mekanismen pakken til høyere lag. Hvis pakken ikke skal av ringen blir pakken lagt i transittbufferet. Det vil være flere transittbuffere for forskjellige prioriteter av datatrafikk på RPR-ringen. En mekanisme som velger pakker fra de forskjellige bufferene (B i figuren) tar så pakken ut fra transittbufferet. Avhengig av prioritet og om pakkene kommer fra sendebuffere eller transittbufferet vil B velge hvilket buffer neste pakke skal velges fra. Denne mekanismen er forklart i utkast 1.0. Hovedtrekkene er at pakker som allerede er på ringen har prioritet over de som skal inn på ringen i denne noden, og pakker med høy prioritet velges før pakker med lav prioritet.

Hovedidéen bak figur 12 på neste side og RPR kalles “buffer insertion ring”. Prinsippet er at transittbufferet skal kunne holde igjen data som passerer i noden til noden har lagt eventuelle data på ringen. La oss tenke oss situasjonen der det skal sendes en pakke med data fra noden. Rett etter noden starter å sende pakken kommer det inn en annen pakke som skal videresendes.



Figur 12: En skisse av en del av en node. Data kommer inn i noden i A. Det vil så avgjøres om hver enkelt pakke skal til denne noden eller videre på ringen via transittbufferet. B velger pakker fra transittbufferet og sendebufferet for å sende videre.

Denne pakken er nødt til å vente til noden er ferdig med å sende pakken som er på vei. Transittbufferet må derfor minst være like stort som den største pakken en node kan sende.

4.4 Datatrafikk på ringen

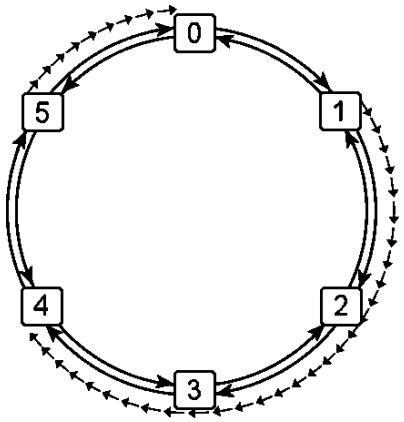
Det skal være tre prioriteter av datatrafikk i RPR, der de to laveste blir behandlet likt når de først har kommet på ringen. Det er to transittbuffere for hver ring i hver node og tre sendebuffere. Mekanismen for valg av hvilket buffer neste pakke skal velges fra velger pakker med høy prioritet fremfor lav prioritet. Mekanismen tar også hensyn til buffere med liten ledig plass. Datatrafikk med lav prioritet blir regulert med hensyn på hvor mye de andre nodene på ringen sender for å fordele RPR-ringens ressurser mellom nodene. Denne mekanismen kalles fairnessalgoritmen som også er en viktig del av RPR standardiseringen.

4.5 Samtidig bruk av ringen, “spatial reuse”

Ideen bak RPR er at man skal få full utnyttelse av ringnettverkets kapasitet på alle måter. I figur 13 på neste side ser vi et eksempel på samtidig bruk av ringen. Pakkestrømmen fra node 1 til node 4 går samtidig som pakkestrømmen fra node 5 til node 0. Pakkestrømmene mellom de ulike nodene går uavhenging av hverandre på de forskjellige ringsegmentene.

4.6 Feilhåndtering / beskyttelse

Nodene i en RPR-ring vil kompensere for feil på ringen slik at meldingsutvekslingen vil fortsette å fungere til tross for feil. Nodene kan ha ulike måter å håndtere feilsituasjoner på som det vil være nyttig for de andre å vite om. Wrapping og styring er to metoder for å håndtere



Figur 13: Figuren viser et eksempel på samtidig bruk av ringen. Data kan gå fra node 1 til node 4 samtidig som det går data fra node 5 til node 0.

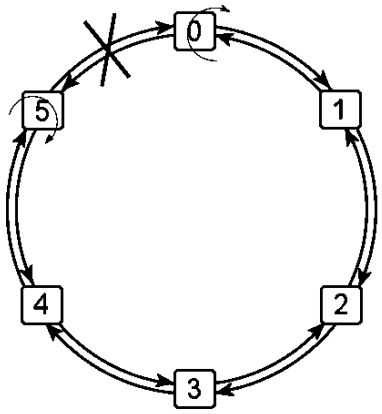
feilsituasjoner. Disse bruker informasjon fra nettverkskartlegging. Wrapping og styring er en del av beskyttelsesmekanismen. Beskyttelse vil bli brukt i denne oppgaven for den engelske betegnelsen “protection”.

Begge måtene å håndtere feilsituasjoner på bruker RPR-ringens muligheter til å nå mottakeren ved å bruke motsatt ring enn normalt. Dette fører til at ved en feil vil ringens totale båndbredde synke. Hvis det er stor last på RPR-ringen vil dette føre til nedsatt gjennomstrømning ved et brudd.

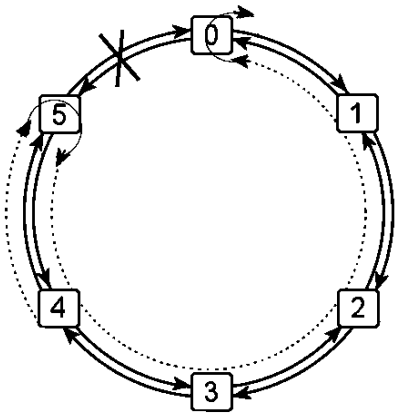
4.6.1 Wrapping

I RPR er det definert to metoder for å rette feil ved fiberbrudd. En av de er wrapping. På figur 14 på neste side ser man at de to linkene mellom nodene 0 og 5 er ødelagt. Pakkene som normalt bruker disse linkene vil da ikke kunne nå den riktige noden over disse linkene lenger. Pakkene som tidligere ble sendt over linkene vil bli wrappet tilbake på den andre ringen (se figur 15 på neste side). Pakkene vil da kunne nå mottakernoden. Wrapping vil skje straks nodene 0 og 5 oppdager feilen. Denne mekanismen skal foregå i maskinvaren, slik at brukeren trenger strengt tatt ikke å vite om feilen. Standarden sier det ikke skal ta lengre tid enn 50ms fra feilen oppstår til pakkene blir sendt en annen vei. Dette intervallet er valgt på grunn av for eksempel sanntidstjenester som går over ringen ikke skal oppleve for lange brudd. Grensen på 50ms vil for telefoni gjøre at partene i samtalen ikke opplever brudd i talen. Wrapping er en valgfri mekanisme i nodene. Hvis wrapping skal være i bruk på RPR-ringen må alle nodene støtte dette [6].

Man kan også se for seg situasjonen der en av de to linkene mellom to noder blir brutt. Det dukker da opp noen problemer som ikke er aktuelle ved bryting av begge de to linkene. La oss si at kun den indre ringen i figur 14 på neste side brytes. Den ytre ringen kan da fremdeles brukes til å sende data mellom node 5 og node 0. Data som skal fra node 0 til node 5 på den indre ringen derimot vil bli wrappet i node 0. Den ytre ringen fra node 0 til node 1 blir da en flaskehals. Data fra node 5 og wrappet data i node 0 vil begge prøve å komme inn på den



Figur 14: Linken mellom node 0 og 5 er ødelagt. Node 0 og node 5 wrapper her pakker.



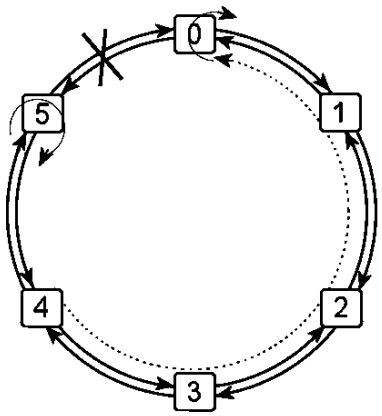
Figur 15: Her ser man hvordan pakkene blir rutet ved en feil.

samme linken. Hvis RPR-ringen skal støtte at kun én av de to linkene blir brutt må det være en mekanisme i nodene som håndterer situasjonen.

4.6.2 Styring

En ulempe med wrapping er at pakkene tar en unødvendig lang vei. På figur 15 på forrige side ser man hvor pakkene går mellom nodene når det er en feil på ringen. En kan se at forbindelsen er brutt mellom node 5 og node 0. Pakkene fra node 4 til node 0 går da en omvei for å komme frem til mottakeren fordi de prøver å nå 0 via node 5.

Man kan også rute pakkene fra node 4 til node 0 den andre veien på RPR-ringen. Dette gir en lengre vei enn hvis RPR-ringen er feilfri, men kortere enn veien ved bruk av wrapping. Dette er mer effektivt med tanke på bruk av ringens ressurser ved at det gir mindre trafikk på ringen. Den nye veien bli da som i figur 16. Denne prosessen kalles “steering” eller styring. Pakkene vil bli styrt i en mer optimal retning. Dette vil føre til at en ny runde med nettverkskartlegging blir startet slik at alle nodene på ringen kan få vite om feilen.



Figur 16: Datarute etter styring eller ny nettverkskartlegging.

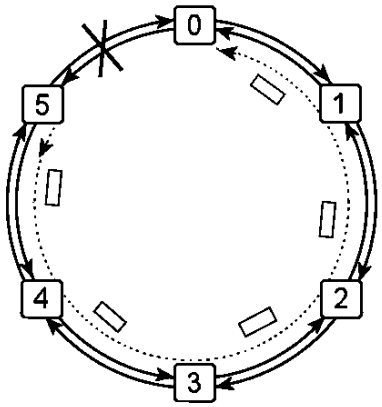
Det er ikke sikkert alle nodene på ringen har mulighet til både wrapping og styring. Som sagt er wrapping en valgfri mekanisme. I så tilfelle kan det være viktig at nodene er klar over de andre nodenes egenskaper. Dermed kan de gjennomføre den mest optimale styringen av pakker. I noen av forslagene som er presentert er det opsjoner i topologipakkene som sier om noden støtter wrapping.

4.6.3 Omstokking av pakker

Sending av pakker i RPR vil vanligvis foregå slik at pakker sendes den korteste veien rundt ringen for å nå den riktige noden. En kunne også tenkt seg en mekanisme der pakker i en datastrøm blir fordelt mellom de to ringene. Det som kan skje da er at pakkene kan komme frem til mottakeren på RPR-ringen i en annen rekkefølge enn de ble sendt. Etter å ha nevnt noen flere situasjoner der pakker kan bli stokket om ser vi hvorfor dette er uheldig.

Hvis det skulle oppstå en feil på ringen slik at en link mellom to noder ikke fungerer vil pakker som vanligvis sendes over denne linken måtte sendes en annen vei. Det er forklart to metoder å gjøre dette på tidligere, wrapping og styring. Wrapping kan brukes til nodene har fått ett nytt topologibilde og kan iverksette styring, eller mekanismene kan brukes uavhengig av hverandre. I det tidspunktet når noden begynner å wrappe pakker vil pakkene gå en lengre vei for å komme til mottakeren. Det er ikke sikkert at pakkene som blir sendt kommer frem i den rekkefølgen de ble sendt. De ulike linkene kan ha forskjellig last som forsinker pakkestrømmen den ene veien. Når noden begynner med styring i steden for wrapping vil pakkene ofte gå en kortere vei. Dette kan føre til at enda flere pakker kan komme i feil rekkefølge til mottakeren. Ved en feilsituasjon der wrapping og styring brukes er det altså mulighet for at pakker blir stokket om to ganger.

Et lignende problem som det over kommer opp når linken som har vært ødelagt blir reparert igjen. Da vil det kunne være en mulighet for å sende pakker en kortere vei enn under feilsituasjonen. Her er det igjen en mulighet for omstokking av pakker hvis noden plutselig begynner å sende pakkene den korteste veien. Dette er lett å se på figur 17. Her vil pakkene fra node 5 til node 0 gå en vei med flere hopp enn den opprinnelige ruten på ett hopp rett fra 5 til 0. La oss da tenke oss at linken mellom node 5 og node 0 blir rettet opp slik at det igjen kan brukes til datatrafikk. Hvis node 5 da umiddelbart begynner å sende pakker ment for node 0 langs den korteste veien vil det være mulighet for at disse pakkene kommer før de sist sendte på den lange veien.



Figur 17: Pakkene går en lengre vei når linken er brutt.

Det er ønskelig å finne en løsning på problemet. Med tanke på blant annet TCP-trafikk som går over RPR-ringen er det ønskelig å ikke stokke om pakker i nettverket. Ytelsen til TCP vil settes ned hvis for mange pakker kommer ut av ordning frem til mottakeren [7].

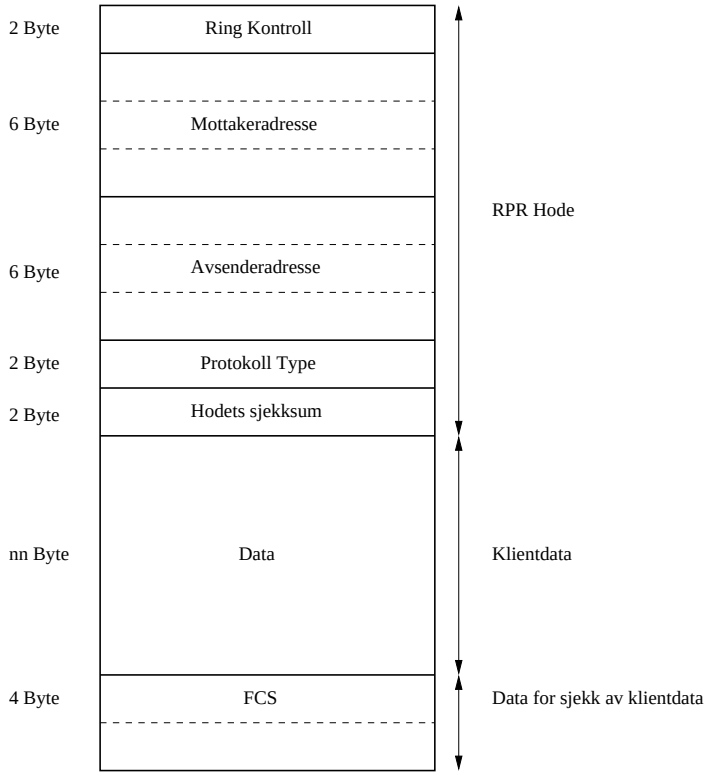
De overnevnte problemene kommer av at nettets topologi forandrer seg. Løsninger på problemene behøver likevel ikke bli løst av nettverkskartlegging, men kan bli løst av andre mekanismer som for eksempel valg av ring.

4.7 Sammenkobling av RPR-ringer

Ringer kan kobles sammen som tidligere vist i figur 9 på side 12. Dette kalles “bridging” eller brosammenkobling. Hvis en node skal sende en pakke videre til en annen ring har den flere muligheter. Sendernoden behøver ikke være klar over hvilken node som skal ta pakken av ringen. I dette tilfellet må noden sende pakken til alle på ringen. Den riktige noden vil da ta pakken av ringen. Noden som binder de to ringene sammen kalles en bro.

Eksempel: Node A2 skal sende en pakke til B2, men vet ikke at det er bro AB0 som er koblet til ring B. A2 må da sende pakken til alle på ringen og AB0 vil ta pakken av ringen. En annen mulighet oppstår hvis sendernoden vet hvilken bro som skal ha pakken. Da vil sendernoden sende pakken direkte til den riktige broa (A2 vil da sende direkte til AB0). Denne problemstillingen er aktuell innen nettverkskartlegging. Nodene kan spre informasjon om seg selv til de andre nodene på ringen ved hjelp av nettverkskartlegging. Det kan være interessant for broene å opplyse om tilkobling til nettet utenfor slik at nodene vet hvilke broer som skal kontaktes hvis mottakernoden ikke er på den samme RPR-ringen. Hver node kan være en bro. Dette kan etterhvert gi mye informasjon slik at en må finne ut hva som er interessant å lagre i hver node. Bridging kan gjøre nettverkskartlegging mer komplisert. Videre problemstillinger rundt bridging vil ikke bli tatt for seg i denne oppgaven.

4.8 Pakkeformat



Figur 18: Figuren viser pakkeformat i utkast 1.0.

Figur 18 på forrige side viser hvordan RPRs pakkeformat ser ut i utkast 1.0. På starten av pakken er det to byte som inneholder diverse kontrollinformasjon. Den første av de to bytene inneholder RPR-pakkens TTL verdi. TTL ("Time To Live") er et heltallsfelt som dekrementeres for hver node som passeres. Bruken av TTL i RPR og denne oppgaven blir presentert i et senere kapittel (kap. 5.1.3). Den andre byten inneholder diverse flagg som indikerer blant annet hva slags type pakke det er og hvilken prioritet den har. Etter kontrollfeltene følger adressene til avsender og mottaker, hver på 48 bit. For at sendernoden skal kunne dirigere pakken til den riktige noden på ringen må hver node ha en entydig adresse eller identifikator. Dette vil lette rutingen av den enkelte pakken på ringen. Utkastet har med mottakernodens og sendernodens identifikator i hver pakke. En mulighet er å bruke MAC-adresser som identifikator. Siden MAC-adresser er entydige vil hver node ha en entydig identifikator på 48 bit. Men det vil neppe være aktuelt med 2^{48} noder på en RPR-ring. Utkastet sier også at maksimal størrelse på en RPR-ring er 255 noder. Derfor kan man diskutere om hver pakke må ha hele denne adressen som sin identifikator eller om det holder med en kortere enn 48 bit. Etter adressene føler protokolltype som blant annet kan fortelle lengden på pakken. Dataene og hodet har hver sin sjekksum.

Nodene må ha en identifikator. Men i tillegg til identifikatoren er det annen informasjon nodene kan ha nytte av å vite om hverandre. De forskjellige nodene kan ha ulike egenskaper. I noen tilfeller kan det være nyttig å ha en oversikt over hva de andre nodene på ringen har mulighet til å gjøre. Til dette kan blant annet topologipakker brukes. Topologipakker vil bli brukt i denne oppgaven som de pakkene mekanismen for nettverkskartlegging sender ut. Avhengig av hvilken mekanisme for nettverkskartlegging det er vil disse pakkene inneholde informasjon om noden selv og eventuelt andre noder. Nodene kan også inneholde informasjon om nodenes muligheter og kapasiteter.

Oppsummering

Vi har her sett på strukturen i en enkel node og hvordan de enkelte nodene opererer på RPR-ringen. RPR-ringen er en "buffer insertion ring" der mottaker tar dataene av ringen og det er muligheter for samtidig bruk av ringen. RPR-ringen er motstandsdyktig mot feil der wrapping og styring er to mekanismer for å gjenopprette datatrafikk mellom noder. Det er også mulighet for å sammenkoble RPR-ringer til å danne mer avanserte nett.

5 Nettverkskartlegging i RPR

Dette kapittelet er delt i tre deler. Den første tar for seg generelle aspekter ved nettverkskartlegging i RPR. I den andre delen vil de to fremlagte forslagene fra utkastet og Darwin bli presentert. De vil bli presentert på et overordnet nivå for å få kjennskap til hvordan de fungerer. Den siste delen vil ta for seg problemer i de fremlagte forslagene. Noe er felles for de to og noe er særegent for de ulike metodene.

5.1 Aspekter ved nettverkskartlegging i RPR

Flere forhold må avklares når det skal lages en mekanisme for nettverkskartlegging i en RPR-ring. Det vil være naturlig å gå ut fra de grunnleggende problemstillingene (se kap. 1.2 på side 2) og jobbe videre fra disse. Nodene skal vite hvilke andre noder som finnes på RPR-ringen, men hvilke egenskaper til de andre nodene de skal kjenne til er også en sentral del av problemstillingen. Foruten de grunnleggende problemstillingene ved nettverkskartlegging dukker det opp flere andre problemer under arbeidet. Disse må tas hensyn til under utvikling av en metode for nettverkskartlegging i RPR.

Det må avklares hvordan alle nodene på ringen skal skaffe seg nettets topologiinformasjon. Det er viktig at informasjonen distribueres til alle nodene uten å skape unødig trafikk på ringen. Hvis det blir veldig mye informasjon som skal utveksles mellom nodene kan lasten dette tilfører RPR-ringen bli unødvendig stor.

RPR skal ikke ha noen “master”. Denne engelske betegnelsen vil her oversettes til norsk som “sjef”. En sjef betegner ofte en som har en annen rolle der andre må forholde seg til sjefen. I RPR betyr dette at det skal være opp til hver enkelt node å gjennomføre nettverkskartlegging ved behov etter et gitt sett med spilleregler. Disse reglene definerer når dette behovet gjøres gjeldende. En RPR-ring uten sjef har flere fordeler. Det vil føre til at det ikke er behov for spesiell programvare eller maskinvare i noen av nodene på ringen. RPR-ringen er heller ikke avhengig av at en spesiell node på ringen hele tiden er i drift for å gjennomføre nettverkskartlegging. Med en sjef på ringen er de andre nodene avhengig av at denne fungerer som den skal. En kan likevel tenke seg at RPR kan ha en mekanisme for nettverkskartlegging som bruker en sjef for å finne ringens topologi. Det finnes også sjef-arkitekturer der en ny sjef velges mellom de gjenværende nodene i nettet hvis sjefen ikke lenger kan oppfylle sitt ansvar [12]. En sjefsløs RPR-ring hindrer derimot ikke at nodene kan samarbeide for å sørge for at alle nodene til enhver tid har et komplett topologibilde av ringen.

5.1.1 Eget buffer for kontrollpakker

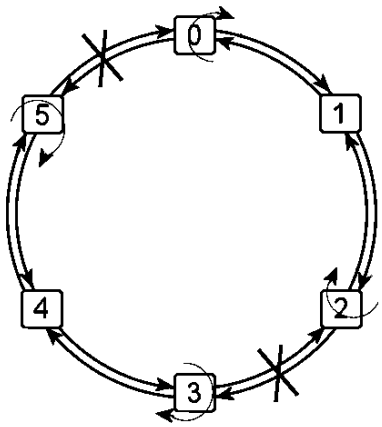
Uansett hvilken metode som brukes for nettverkskartlegging så må pakkene mekanismen skal sende legges inn i et buffer for å senere legges på ringen. I de tidlige versjonene av utkastet var det ikke noe eget buffer for kontrollpakker, noe som førte til at disse måtte konkurrere med datapakker i sendebufferene. Senere utkast, blant annet 1.0, har eget buffer for RPR kontrollpakker. Kontrollpakkene konkurrerer nå med datapakker når det skal legges en pakke på RPR-ringen. Dette vil håndteres av mekanismen som velger pakker fra de ulike bufferene.

5.1.2 Grenser til andre mekanismer i RPR

5.1.2.1 Beskyttelse Det er flere mekanismer i RPR der grensene mellom disse og nettverkskartlegging er uklar. Beskyttelse, som implementerer RPR-ringens motstandsdyktighet mot feil [6], er eksempel på dette.

Nettverkskartlegging og beskyttelse vil i noen tilfeller måtte jobbe sammen for å sørge for å holde grensen på 50ms fra en feil oppstår til en alternativ forbindelse er oppe igjen. Forbindelsen vil da ikke gå den mest optimale veien, men den mest optimale med den feilen som har oppstått. Når en node mottar en beskyttelsemelding som indikerer en brutt link kan noden begynne å sende pakker den andre veien da dette fører til en mer optimal retning enn eventuell wrapping. Noden kan da bruke informasjonen i topologibildet. Ved å bruke denne informasjonen vil noden raskt kunne finne en ny vei til de nodene som blir berørt. De berørte nodene er da de nodene som tidligere ble nådd forbi det punktet som nå er brutt. Hvilke noder som blir berørt er forskjellig i de ulike nodene på RPR-ringen. I figur 17 på side 20 vil det for node 5 etter bruddet være en kortere vei til node 0 og 1 langs den indre ringen. Før bruddet var den korteste veien til node 0 og 1 langs den ytre ringen. For node 4 derimot vil det kun være node 0 som nå har en kortere vei langs den indre ringen.

Det kan oppstå flere feil på ringen samtidig og da er det viktig for nodene å vite at man ikke helt uten videre kan sende pakker den andre veien på ringen og håpe at de kommer frem til den rette mottakeren. Ved en dobbel feil på en RPR-ring vil det oppstå to isolerte deler av nettet som vil fungere uavhengig av hverandre. Hvis det da enda skal sendes data mellom nodene kreves minst to noder på hver av de isolerte delene. Figur 19 viser situasjonen som kan oppstå.



Figur 19: Figuren viser en RPR-ring med to feil som gir to isolerte deler av nettet.

5.1.2.2 Valg av ring Alle pakker som sendes ut på RPR-ringen har en mottakeradresse. Pakken skal tas av ringen i en eller flere av nodene på RPR-ringen. Når disse pakkene skal sendes må avsendernoden velge ut fra gitte kriterier hvilken ring pakkene skal legges på. I RPR er det to muligheter ettersom det er to ringer. De to ringene har i de fleste tilfeller forskjellig avstand til mottakernoden. Slik nettverk i hovedsak fungerer i dag er ruting mellom

noder i nettet basert på antall hopp. Hvis det en vei er 2 hopp til en mottaker, og en annen vei på 3 hopp så vil veien med 2 hopp ofte velges uten tanke på linklengder. Hvis det er flere mottakere av samme pakke vil valg av ring kompliseres i forhold til hvis det er én mottaker. Med flere mottakere av en pakke langs RPR-ringen må noden bestemme seg for hvordan denne pakken skal distribueres til mottakerene på den mest effektive måten.

Standarden setter få krav til valg av ring. Det kreves kun at nodene skal velge samme ring for pakker som skal til samme mottakeradresse. Dette er for at en flyt av pakker ikke skal bli reordnet på vei gjennom ringen. Hvis pakkene i en og samme flyt blir rutet forskjellige veier vil pakkene kunne "ta igjen" pakker som har gått en annen vei.

Valg av ring i RPR er ikke direkte en del av nettverkskartlegging, men denne mekanismen bruker topologibildet for å velge ring. Valg av ring kan konfigureres av en administrator slik at data til en mottaker velger den ringen som administratoren ønsker. Hvis dette ikke er gjort må RPR gjennomføre den mest optimale fordelingen av pakker på de riktige ringene. På grunn av at valg av ring avhenger av topologibildet er det viktig at strukturen på topologibildet er slik at nodene lett kan finne den raskeste veien på ringen til mottakernoden. Det finnes flere måter å bygge opp et topologibilde på som forklart i kapittel 3.2.1 på side 11.

5.1.3 TTL i RPR

I RPR er TTL et 8 bit langt felt i RPR-pakkehodet. Feltet hindrer pakker som går på ringen å sirkulere evig. Feltet er et heltall som dekrementeres i hver node pakken passerer på vei til mottakeren. Utkast 1.0 sier at unicastpakker (pakker med 1 mottaker) skal ha TTL verdi i utsendte pakker satt til avstanden til mottaker lagret i topologibilde. Topologipakker må derimot basere seg på at de ikke vet størrelsen på ringen. Utover i denne oppgaven blir TTL behandlet i ulike situasjoner. Tidspunktet for når verdien av feltet blir oppdatert under prosessering i nodene er viktig. Dette har betydning for utregninger der TTL blir brukt. Denne oppgaven baseres på at feltet i topologipakker blir dekrementert før pakkene blir prosessert av mekanismen for nettverkskartlegging. Sirkulering hindres ved at pakker som har et felt som blir 0 etter dekrementering blir prosessert i noden men ikke sendt videre langs ringen.

5.2 Presentasjon av de fremlagte forslagene

Dette kapittelet tar for seg to av de metodene for nettverkskartlegging som er presentert for arbeidsgruppen. De to er forslaget fra Darwin og utkastet.

Av de fremlagte dokumentene er det Darwin og forslaget i utkast 0.3/1.0 som er testet. Disse to er valgt på grunn av at de presenterer de to forskjellige metoder med ulikt hovedprinsipp å gjennomføre nettverkskartlegging på. Andre forslag som er presentert kan være litt annerledes enn disse men grunnidéen vil bygge på samme prinsipp som enten Darwin eller utkastet. Det er mangler i de to forslagene jeg har tatt for meg. Darwin er et forslag som kom frem tidlig i standardiseringsperioden og har derfor ikke blitt fullstending utviklet. Utkastet er, som ordet sier, enda bare et utkast, noe som gjør at denne mekanismen også har mangler. Noen av manglene er deler av mekanismen som trengs ved utvikling av simulatormodellen. Manglene er tatt opp som problemer under implementering da de som regel har kommet opp underveis. Jeg har også bygd opp og testet min egen mekanisme for nettverkskartlegging.

Denne blir presentert i kapittel 7.

De to prinsippene i Darwin og utkastet:

- Det ene prinsippet er at nodene skal samle informasjon om de andre nodene for å bygge opp sitt eget topologibilde. Det foregår slik at mekanismen for nettverkskartlegging sender regelmessig, eller ved behov, ut topologipakker på begge ringene der de andre nodene legger til informasjon om seg selv. Informasjonen om alle de andre nodene blir så prosessert i avsenderen i sin helhet for å bygge opp topologibildet. Metoden kalles også lenking. Dette er idéen Darwin bygger på.

- Det andre prinsippet er slik at nodene på RPR-ringen gir bort informasjon om seg selv i topologipakkene de sender ut for at de andre nodene skal få bygd opp sitt topologibilde. Det vil da foregå slik at nodene langs ringen prosesserer pakkene ettersom de kommer inn. Topologibildet vil da bli bygd opp litt etter litt helt til noden har mottatt pakker fra alle de andre på RPR-ringen. Utkastet bruker denne idéen for å gjennomføre nettverkskartlegging.

Videre vil de to fremlagte forslagene bli grundigere forklart.

5.2.1 Darwin

Darwin er en videreføring av Gandalf [1] forslaget. Darwin tar også elementer fra Alladins [4] forslag for nettverkskartlegging. Mekanismen for nettverkskartlegging i Darwin er mangelfull. Det blir presentert elementer som ikke er forklart, men hovedprinsippene kommer tydelig frem.

Nodene på RPR-ringen vil i alle topologipakker legge til informasjon om seg selv. I dataene hver node tilføyer pakken ligger informasjon om nodens MAC-adresse og to byte MAC-tilleggsinformasjon som er vist i tabell 1. MAC-tilleggsinformasjon og MAC-adressen kalles samlet MAC-bindingene.

Bit	**Verdi**
0	Enkelt sendebuffer(0) / Dobbelt sendebuffer
1	Ring identifikator (indre 1 / ytre (0)
2	Wrapped node (1) / Ikke wrapped node (0)
3	Mulighet for wrapping
4-6	Versjon av fairness melding
7-13	Vekt
14-15	Reservert

Tabell 1: Dette er MAC-tilleggsinformasjonen hver node legger til topologipakkene. Tabellen er tatt fra forslaget [2].

Forslaget holder av 7 bit til et vekt felt som ikke er forklart i forslaget. Flere av de andre feltene i tilleggsinformasjonen er heller ikke forklart men kan brukes i andre mekanismer som beskyttelse og fairness. Fairness er en mekanisme som sørger for at alle nodene på RPR-ringen får sin del av nettets båndbredde.

Når de to pakkene, en på hver ring, har gått rundt ringene vil de bli prosessert i avsendernoden for å finne et topologibilde av RPR-ringen.

5.2.2 Utkastet

Også i dette forslaget sender alle nodene ut topologipakker regelmessig eller ved behov. Disse pakkene inneholder nodens egen MAC-adresse i tillegg til adressene til nodens to naboer. Hvis noden ikke vet adressen til sine naboer indikeres dette med “0” som naboadresse. Noder på RPR-ringen vil prosessere disse pakkene etterhvert som de går rundt ringen. Avsendernoden blir lagt til i topologibildet med informasjon om dens naboer. Hvis en node mottar en pakke der naboadressene er satt til “0” vil den utløse sending av topologipakker. Slik vil noden som indikerte “0” som nabo få informasjon om de andre nodene på ringen.

5.3 Problemer i de fremlagte forslagene

Forslagene som er lagt frem er ikke komplette. Mange sider ved utviklingen av en komplett metode for nettverkskartlegging er ikke dekket av det som kommer frem i forslagene. Dette kapittelet tar for seg noen av problemstillingene som har kommet opp basert på de fremlagte forslagene. Det starter med noen generelle punkter som er felles for de ulike forslagene og fortsetter med å ta for seg de to forslagene som er presentert. De to som er valgt er valgt på grunn av at måten de løser nettverkskartlegging på er veldig forskjellig, og at disse to tilsammen viser en stor del av mulighetene man har ved utvikling av en mekanisme for nettverkskartlegging.

5.3.1 Sjekk av topologibildet

Som en del av mekanismen bør det være mulig for nodene å sjekke om topologibildet er korrekt. Det kan være flere måter å gjøre dette på utifra hvilken av de ulike metodene for nettverkskartlegging som brukes. Noen av mekanismene vil ikke kunne støtte en slik test av topologibildet på grunn av at hver post i topologibildet er isolert slik at det er umulig å sammenligne med en annen verdi. Et eksempel på fysisk umulig topologi er hvis en node har en annen node 3 hopp unna seg selv, men ikke har en annen 2 hopp unna. Et topologibilde kan midlertidig være i en slik tilstand hvis det har skjedd en endring i nettets topologi. Nodene kan også samarbeide for å finne ut om deres topologibilde er konsistent. Hvis en node 1 ser 2 som sin nabo på den ytre ringen, og node 2 ser node 3 som sin nabo på den indre ringen der 2 skulle sett 1 er det en feil. Dette eksempelet krever da at nodene finner sine naboer i det lokale topologibildet og kan distribuere denne informasjonen rundt til de andre nodene på ringen.

5.3.2 Fjernet node fra ringen

Et problem som gjelder for flere av metodene for nettverkskartlegging er at topologipakkene må før eller siden tas av ringen. Dette gjøres oftest av avsenderen av topologipakkene. Et problem oppstår da hvis denne noden fjernes fra ringen eller feiler slik at pakkene sirkulerer til TTL blir 0. Pakkene vil til slutt bli borte fra ringen, men hva vil skje med mekanismen nettverkskartlegging når den samme pakken kommer til de samme nodene flere ganger? Ettersom hvilken metode som brukes vil det da kunne føre til inkonsistens i topologibildet til nodene. I utkast 0.3 [5] er avstanden til nodene basert på TTL-verdi i pakkene. Hvis det er pakker som sirkulerer på ringen vil nodenes mekanisme for nettverkskartlegging prosessere disse flere ganger. Dette må tas hensyn til under implementeringen av mekanismen siden det ikke nevnes i utkastet.

5.3.3 Koordinert nettverkskartlegging

Nodene i RPR-ringen kan gjennomføre nettverkskartlegging regelmessig eller ved behov. Behovet kan gjøres gjeldene når noden selv finner ut at en endring i nettets topologi har funnet sted, eller når en av de andre nodene har oppdaget en endring. I det siste tilfellet vil noden som oppdaget feilen bli nødt til å spre informasjon til de andre nodene på ringen om at det har skjedd en endring slik at de andre kan få oppdatert sitt topologibilde. Ikke alle de fremlagte forslagene har en slik koordinert mekanisme, men baserer seg på at hver enkelt node skal gjennomføre nettverkskartlegging uten hjelp av andre. Det er flere fordeler med en koordinert mekanisme som gjør at dette er ønskelig i nettverkskartlegging. Hvis det skjer en feil på ringen der de nærmeste nodene oppdager feilen først vil disse kunne si ifra til de andre nodene at nettverkskartlegging må gjennomføres. Slik vil alle nodene raskt kunne få et oppdatert topologibilde av ringen etter feilen / endringen. Hvis denne koordinerte mekanismen ikke var tilstede ville nodene bli nødt til å vente på en eventuell timer som gikk ut før de gjennomførte sin regelmessige 'sjekk' av ringen for å se om deres topologibilde var rett. Det mest ekstreme tilfellet ved en slik situasjon er at en node må vente hele perioden for topologitimeren før sender ut topologipakker og får oppdatert sitt topologibilde.

5.3.4 Spredning i tid

Ved bruk av regelmessig utsending av topologipakker er det viktig at alle nodene ikke sender samtidig. Dette er for å unngå at topologipakker hoper seg opp i nodene og bruker mer nettverksressurser enn nødvendig. Mekanismen i de forskjellige nodene kan ha samme periodetid for sin topologitimer. Men selv om nodene har samme periodetid er det veldig liten sannsynlighet for at alle nodenes mekanisme for nettverkskartlegging starter samtidig. Slik vil dette problemet ikke være et problem fordi det vil løse seg selv. Hva timerperioden skal settes til er ikke fastsatt i utkastet. Vi vil se utover i oppgaven at denne verdien med fordel ikke er en fast verdi.

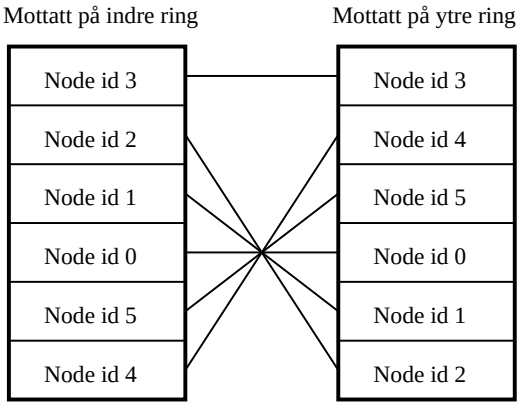
5.3.5 Darwin

Det er ikke foreslått en metode i Darwin for å bygge opp topologibildet. Dermed blir dette opp til implementasjonen. Delkapittel 6.2.2.1 på side 38 tar for seg hvordan jeg har bygget

opp topologibildet for Darwins metode.

Det er ikke foreslått noen tid for hva som skal være timerperiode for regelmessig nettverkskartlegging. Jeg baserer meg på grensen som er satt for 50ms innen en ny rute skal være satt opp ved feil. Da er det viktig å finne ut hvor lang tid denne mekanismen bruker for å oppdage ny topologi. Dette kan avhenge av hvilken verdi man setter på den regelmessige sjekkingen av topologibildet.

Det siste forslaget til Darwin mener at topologibildet ikke skal endres før det kommer to pakker som tilsier at topologibildet er annerledes enn det topologibildet som er lagret. Dette er litt uklart for det kommer ikke frem hvilke pakker som telles i denne sammenheng. Det er to muligheter. Den ene er at det må komme to pakker som er annerledes langs hver av ringene, altså at det må komme fire pakker totalt før topologibildet endres. Den andre muligheten er å sammenligne pakkene som kommer inn fra de to ringene, og hvis begge antyder en endring i nettets topologi så skal topologibildet endres. De to pakkene kan sammenlignes fordi rekkefølgen på MAC-bindingene i de to pakkene er like men speilvendt, som vist i figur 20.



Figur 20: Figuren viser innholdet i de to topologipakkene node nummer 3 får tilbake når de har gått en runde rundt ringen (kun nodens identifikator er vist da det kun er denne som er interessant her).

Node nummer 3 er valgt kun som et eksempel. Det vil være likt for alle noder. Vi ser i figuren at node nummer 3 selv har lagt inn den første posten i begge topologipakkene. Uten MAC-bindingene til avsender selv er den første posten i den ene pakken lik den siste posten i den andre pakken. Det kan hende at en av pakkene kommer tilbake til avsender med en samling av MAC-bindinger som ikke gjenspeiler nettets virkelige topologi. Dette kan skje hvis det for eksempel skjer en uforutsett feil i en av nodene akkurat da topologipakken passerer. Denne feilen kan skyldes bitfeil eller midlertidig feil i topologibildet.

Hvis metoden som krever to pakker per ring før en endring i topologibildet brukes vil i værste fall følgende kunne skje. Rett etter en node har kjørt en runde med nettverkskartlegging forlater en node RPR-ringen slik at nettets topologi forandrer seg. Noden er da nødt til å vente to hele perioder for nettverkskartlegging før topologibildet i denne noden har forandret seg. Hvis man skal basere seg på 50ms grensen for sanntidsapplikasjoner er man nødt til å ha en periode som er mindre enn 25ms for sending av topologipakker.

Hvis en node på RPR-ringen forlater nettet av en eller annen grunn kan det være mulighet for at det er pakker på ringen ment for denne noden. Dette kan også være tilfelle for topologipakker som noden selv har sendt ut rett før den gikk ned. Det må finnes en mekanisme på RPR-ringen som hindrer disse pakkene å sirkulere i evig tid. TTL verdi i pakkene vil kunne hindre dette og vil kunne fungere på Darwins mekanisme for nettverkskartlegging. Det eneste som vil skje er at alle nodene som fremdeles er på ringen vil fortsette å legge på sine MAC-bindinger helt til TTL er lik 0. For at disse pakkene ikke skal bli altfor store er det viktig at TTL settes til en fornuftig verdi slik at de når rundt ringen til avsenderen, men samtidig ikke sirkulerer for mye ved en feilsituasjon. Derfor er valg av TTL en viktig avgjørelse. Ved utsendelse av topologipakker skal man ikke vite hvor mange noder det finnes på ringen, eller om noen av disse nodene wrapper pakker. TTL feltet kan være stort nok til å nå alle noder på ringen med en feilfri ring, men hvis det foregår wrapping vil avstanden tilbake til seg selv bli betydelig lenger enn i en feilfri situasjon. Avstanden til seg selv ved wrapping vil bli (2*(N-1)) hopp der N er antall noder på ringen. Det er umulig å vite om ringen er wrappet eller ikke. Det er derimot mulig å anslå hvor mange noder som finnes på ringen ut i fra topologibildet. Ved oppstart har ikke noden noe topologibilde å gå utifra, men kan da basere TTL verdi i topologipakker på maksimalt antall noder på ringen. Senere kan hver node basere utsendt TTL i sine pakker på hvor mange noder som finnes i topologibildet. For at pakkene skal kunne komme tilbake til avsender ved en eventuell feilsituasjon må verdien justeres med formelen over. Senere utkast forslår å ikke telle ned TTL når pakker er på den motsatte ring enn den de ble sendt på for å unngå problemet over. Dette har også med størrelsen på TTL feltet og gjøre i forhold til maksimal ringstørrelse.

5.3.5.1 Konsistent topologibilde Det er viktig at innholdet i topologibildet stemmer med virkeligheten. Det kan da være hensiktsmessig å ha en mekanisme som sjekker innholdet i topologibildet og finner ut om topologibildet er konsistent. Vi har tidligere sett at man kan sammenligne de to pakkene som blir sendt på hver ring når de kommer tilbake til avsendernoden. Hvis de er like kan bildet bygges opp. Det kan også være interessant å sjekke informasjonen som ligger i topologibildet for konsistens. Dette vil kreve at bildet bygges opp på en slik måte at dette er mulig. Darwin nevner ingenting om en slik mekanisme for å sjekke det lagrede topologibildet. Siden topologibildet i Darwin bygges opp med den informasjonen som finnes i de to topologipakkene og pakkene kan sammenlignes for konsistens vil en ytterligere konsistenssjekk av bildet måtte baseres på annen informasjon. Redundant informasjon i pakkene (informasjon som ikke trengs for å få et komplett bilde) kan også brukes for å sjekke topologibildet.

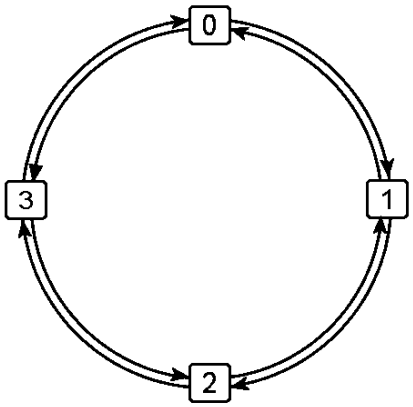
5.3.6 Utkastene

5.3.6.1 Fjerning av en node Forslaget som er lagt frem i dokumentet er ikke helt komplett. Forslaget tar ikke for seg hvordan man skal fjerne en node fra topologibildet etter en node har blitt fjernet fra ringen. Problemet gjøres gjeldende i tilfeller der det fjernes en eller flere noder fra ringen. Noen av nodene som er igjen vil da fremdeles kunne ha nodene som har blitt fjernet i sitt topologibilde.

Det er flere løsninger på dette problemet. Nodene kan slette hele sitt topologibilde når den oppdager en endring i nettets topologi for å så starte på nytt for å få oppdatert topologibildet.

Dette vil gi en periode uten noe topologibilde i nodene. Da sending av pakker i nodene er avhengig av et topologibilde vil sending av pakker ikke være mulig i denne perioden. Nodene kan også bygge informasjonen i topologibildet ut fra antall noder som finnes på ringen. Dette antallet måtte baseres på TTL i sine egne topologipakker som noden sender ut på ringen. Hver enkelt node kan da finne ut på grunnlag av TTL i utsendt pakke og mottatt pakke hvor mange noder som finnes på ringen (n = TTLiUtsendt - TTLiMottatt), der n er antall noder på ringen. For at dette skal fungere kreves prosessering av TTL-felt i henhold til kapittel 5.1.3 på side 25. Antall poster i topologibildet som er gyldige vil da bli de n første i tabellen. Dette vil føre til at nodene ved siden av noden som har blitt fjernet fra ringen ikke vil kunne bruke den siste posten i sitt topologibilde der den fjernede noden fremdeles er.

En lignende feil vil også gjøres gjeldende i de andre nodene som ikke er nabonoder ved fjerning av en node. Alle nodene vil da ha en post for mye i sitt topologibilde der de har en av de andre nodene i sitt topologibilde flere ganger. Dette er vist i figur 21 og figur 22 på neste side. I figur 21 har nodene bygget opp et topologibilde av ringen med de tre andre nodene. Begge figurene viser topologibildet til node 0 for begge ringene. Senere, i figur 22 har node 3 blitt fjernet fra RPR-ringen, noe som de andre nodene må oppdage ved neste runde nettverkskartlegging. Rad to og tre i tabellen vil bli skrevet over av de pakkene som kommer inn fra node 1 og 2. Den første raden er noden selv. Men rad 3 i tabellen, posten som skal betegne en node 3 hopp unna, vil ikke bli skrevet over av noen nodes pakker på grunn av at det ikke finnes noen lenger tre hopp unna.

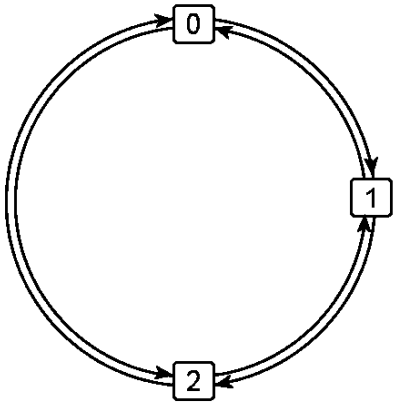


Antall hopp	Ytre ring	Indre ring
0	0	0
1	1	3
2	2	2
3	3	1

Figur 21: Denne figuren viser en ring med 4 noder og topologibilde som node 0 vil bygge opp basert på informasjon den får fra de andre nodene.

For å løse problemet med den ekstra posten kan alle nodene slette hele topologibildet. Dette vil kunne føre en periode der RPR-ringen ikke har mulighet for å sende pakker og er ikke en optimal løsning. Løsningen nevt tidligere med å kun bruke et bestemt antall poster i topologibildet vil også fungere på det skisserte problemet.

5.3.6.2 Sirkulering av pakker på ringen Dette utkastet er sårbart for feilen nevnt i punkt 5.3.2 på side 28 om sirkulerende pakker på ringen hvis en node blir fjernet etter å ha sendt ut sine topologipakker.



Antall hopp	Ytre ring	Indre ring
0	0	0
1	1	3
2	2	2
3	***3***	***1***

Figur 22: Her er den ene noden fjernet fra den virkelige topologien, men node 0 og de andre nodene må ha en måte å fjerne den siste posten i topologibildet på.

5.3.6.3 Utløsing av pakkesending Utkast 0.3 gir motstridende informasjon om utløsing av pakkesending på RPR-ringen. Alle noder skal sende ut en topologipakke på hver ring hvis de merker at det har kommet en ny node inn på ringen. Dette vil indikeres med naboadresse satt til 0 i pakkene den nye noden sender ut. Ved tillegging vil det være en ny node som kan iverksette denne prosessen. Ved fjerning av en node vil det ikke være slik. Utkastet sier at nodene skal utløse utsending av topologipakker hvis de merker at sin nabo har endret seg. Det er derimot ingen mekanisme som vil føre til at alle de som ikke er naboer vil utløse pakkesending. Dette vil føre til at avstandene lagret i bildet vil bli feil inntil timerperioden for nettverkskartlegging vil gå ut.

En mulig løsning på dette problemet vil være at nodene som merker at det har skjedd en endring sender ut topologipakker med nabonoder satt til 0. Man vil slik bruke den samme delen av mekanismen som brukes for tillegging av en node.

Oppsummering

Ved utvikling av en mekanisme for nettverkskartlegging må man ta hensyn til blant annet de grunnleggende problemstillingene. Grenser til andre mekanismer, som for eksempel beskyttelse og valg av ring, må fastsettes.

De to fremlagte metodene, Darwin og utkast 0.3/1.0 bygger på to ulike prinsipp. Darwin samler inn informasjon fra de andre nodene, mens utkastene gir informasjon som seg selv til de andre nodene.

Mekanismene er ikke komplette. Det vil derfor være aspekter ved de ulike som er uklart hvordan skal løses. Hvordan skal topologibildet sjekkes for konsistens og hva skjer med sirkulerende topologipakker? Dette er bare noen av spørsmålene. Det er også problemer som er særegne for de ulike metodene siden de bygger på forskjellig prinsipp.

6 Simulering og drøfting av de fremlagte forslagene

Dette kapittelet vil begynne med å ta for seg simulatorteori og hvordan den simulatoren som er brukt i denne oppgaven fungerer. Hendelsene som skal simuleres vil bli gjennomgått. Deretter følger selve simuleringen for de to fremlagte forslagene. For hver av de to er hypoteser, resultat og drøfting hoveddelene i kapittelet. Som forklart tidligere skal vi senere presentere, simulere og drøfte min egen metode for nettverkskartlegging.

6.1 Innledning

For å kunne sammenligne metodene for nettverkskartlegging har jeg videreutviklet en simulator skrevet i hovedsak av Stein Gjessing ved Simula Research Laboratory. Simulatorkjernen er skrevet av Olav Lysne, også ved Simula Research Laboratory. Simulatoren er skrevet i programmeringsspråket Java. Simulatoren er et program for å simulere datatrafikk og hendelser på et datanettverk som i dette tilfellet er RPR.

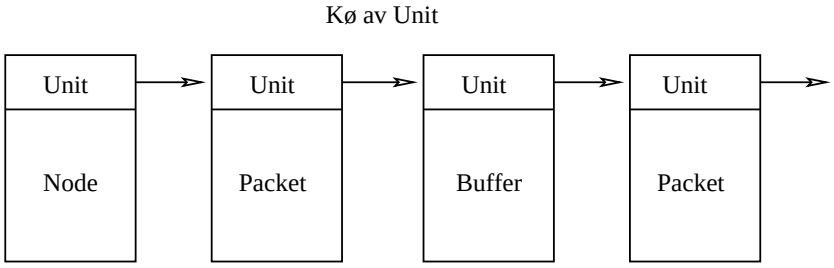
6.1.1 Simulatorteori

Simulatorkjernen er bygd opp slik at den danner en kø av hendelser. Hver hendelse i simulatorkøen har et tidsstempel. Simulatoren behandler hver hendelse etter tur og starter med hendelsen med lavest tidstempel. Tidstemplene skal gjenspeile virkelig tid, men prosessering av en hendelse i denne simulatoren tar null simulatortid. Dette fører til at kjøringer av simulatoren med de samme parametere vil føre til samme resultat uavhengig av prosessorkraft. Simulering av at en hendelse tar tid må kodes eksplisitt. For eksempel når en datapakke i simulatoren blir lagt inn i et buffer skjer det to hendelser i simulatorkøen. En når pakken starter sin ferd inn i bufferet, og en da pakken er helt inne i bufferet. Denne andre hendelsen blir lagt i simulatorkøen under prosesseringen av den første hendelsen. Et tilsvarende eksempel i denne simulatoren er sending av pakker langs en link mellom to noder. Det gjøres prosessering i avsendernoden når pakken skal sendes ut på linken. Etter denne prosesseringen er gjort vil det bli lagt inn en hendelse i simulatorkøen som "vekker opp" simulatoren når pakken har nådd mottakeren. Man vil da ha simulert hendelsen der pakken går over linken.

En simulator som modellerer en kø av hendelser på denne måten kalles en "diskret hendelsesimulator". Kun hendelser som er relevante for den aktuelle simulatormodellen modelleres og vil som følge av dette legges inn i simulatorkøen under kjøring og prosesseres. Som i eksempelet over trengs det å prosessere en pakke når pakken er i de forskjellige nodene. Andre hendelser kan være mindre interessante og kan da utelates fra simulatorkøen. Hendelser som ligger i simulatorkøen vil bli behandlet og vil da bruke prosesseringskraft. Desto færre hendelser i køen desto kortere tid tar simuleringen.

Alle objektene i simulatorkøen implementerer en metode definert i en abstrakt superklasse. Simulatorkjernen kaller denne metoden for å utføre operasjonene på implementasjonen av det spesifikke objektet. I denne simulatormodellen heter den abstrakte superklassen "Unit" og subklassene kan være blant annet av type "Node", "Packet" og "Buffer". Alle disse klassene implementerer en metode som kalles "act()" som er definert i "Unit" klassen. På grunn av oppbygningen der "act()" er definert i superklassen trenger simulatoren kun å ha referanse til

et "Unit" objekt. Simulatoren trenger ikke å vite hvilken type dette objektet er av. Figur 23 illustrerer tankegangen.



Figur 23: Figuren illustrerer hvordan simulatoren bygger opp en kø av "Unit" der hvert objekt i køen er en subklasse av Unit. Subklassene kan være for eksempel av type "Node", "Packet" eller "Buffer".

6.1.2 Bruk av simulatoren

Før jeg begynte videreutviklingen av simulatoren var den i bruk for testing av en fairness-algoritme på RPR-ringen. Det var satt opp en RPR-ring med en fast topologi på 16 noder. Disse nodene sendte pakker til hverandre for å simulere datatrafikk og teste fairness-algoritmen.

6.1.3 Tester

Jeg skulle bruke simulatoren for å evaluere og sammenligne ulike metoder for nettverkskartlegging. For å sammenligne metodene vil simulatormodellen av RPR-ringen påføres hendelser som endrer nettets topologi. Dette vil føre til at mekanismen for nettverkskartlegging må finne denne nye topologien i alle nodene. Hendelsene som påføres skal modellere hendelser som kan skje på en virkelig RPR-ring. Det må da settes sammen en mengde tester som tilsammen dekker over flest mulig av de hendelsene som kan skje på en virkelig RPR-ring. Ved å kjøre alle disse testene på de utvalgte mekanismene for nettverkskartlegging vil resultatene fra de ulike kunne sammenlignes. Testene som følger er satt sammen på det grunnlag av at de skal dekke over flest mulig av de hendelsene som kan skje. Testene er drøftet ytterligere etter de er presentert.

Mekanismene ble testet med med testene i tabell 2 på neste side. Tabellen tar for seg testenes overordnede mål og det er også tabellen som vil bli referert til senere i oppgaven. Testene er grundigere forklart i egne delkapitteler.

Test nr.	Testbeskrivelse	Variasjoner
1	Mål tid fra oppstart av simulatoren til alle nodene har et komplett topologibilde av ringen.	Prøv med forskjellig antall noder på ringen og se om tiden endres lineært.
2	Gjør test 1 om igjen, men ta bort en node halvveis i tiden målt i 1	
3	Ta bort en node etter at nettets topologi har stabilisert seg. Mål tiden til alle har oppdatert sitt topologibilde.	Varier tiden for regelmessig nettverkskartlegging for de metodene dette gjelder.
4	Legg til en node etter stabilisert topologi og mål tiden for hvor lang til det har tatt før alle nodene har fått et konsist topologibilde av ringen.	Varier tiden for regelmessing nettverkskartlegging og finn ut om det blir endringer i tiden målt.
5	Ta bort en node og legg til en node samtidig (endre identifikator på en node).	
6	Ta ned en link etter stabilisert topologi. Bruk wrapping og styring for å få et nytt topologibilde av ringen. Mål tid fra endring av topologi til alle nodene har oppdatert sitt topologibilde.	
7	Bryt to linker for å lage to separate nett. Observer hvordan nettverkskartlegging takler dette.	
8	Sett de to nettene i #7 sammen igjen og observer hvordan nettverkskartlegging takler dette.	

Tabell 2: En oversikt over testene som kjøres på mekanismene.

6.1.3.1 Oppstart - test 1 Når simulatoren starter opp og setter opp en RPR-ring vil alle nodenes mekanisme for nettverkskartlegging sende ut topologipakker for å oppdatere sitt eget bilde.

Når størrelsen på ringen øker vil antall noder og en rundetid øke proporsjonalt.

6.1.3.2 Fjernet node ved oppstart - test 2 I denne testen skal en node fjernes. Tiden noden skal fjernes på er halvparten av tiden målt i test nummer 1. Dette tidspunktet er valgt kun for å sjekke hva som skjer dersom man fjerner en node midt i en runde nettverkskartlegging.

6.1.3.3 Fjerning av node - test 3 Fjerning vil gjøres slik at linkene ut fra en node vil gå forbi nabonoden og peke på noden 2 hopp bortenfor på RPR-ringen. Før endringen skjer skal alle nodene ha et rett bilde av nettets topologi. Mekanismen skal ha gått til alle nodene har fått topologibilde. Hvis nettverkskartlegging er bygget slik at utsending av topologipakker blir spredt etter hvert skal noden fjernes når utsending er relativt jevnt fordelt i tid.

6.1.3.4 Lagt til node - test 4 Noden vil settes inn på RPR-ringen mellom to vilkårlige noder.

6.1.3.5 Endring av identifikator - test 5 En node på RPR-ringen får en ny identifikator som ikke finnes på RPR-ringen fra før. Dette vil skje på null tid slik at nodene må fjerne den gamle identifikatoren fra topologibildet og legge til den nye.

6.1.3.6 Bryting av en link - test 6 Linkene på begge ringene mellom to noder brytes. De to nodene vil indikere dette ved å wrappe pakker slik at ingen pakker traverserer linkene lenger.

6.1.3.7 Bryting av to linker - test 7 Samme som forrige test, men det skjer to steder på RPR-ringen samtidig.

6.1.3.8 Sammensetting av brutt nett - test 8 Alle brutte linker vil rettes opp slik at data kan gå den raskeste veien på RPR-ringen.

Det er tatt med noen tester som er interessante for å se hvordan nettverkskartlegging fungerer, men som det ikke er veldig sannsynlig at vil skje på et virkelig RPR-nettverk. Dette gjelder de to første testene i tabell 2. Det er mindre sannsynlig at mekanismen for nettverkskartlegging i alle nodene på RPR-ringen blir startet samtidig. Test 2 baserer seg i hovedsak på test nummer 1, men er mere reell fordi den kan gjenspeile andre situasjoner som kan oppstå. Jeg fjerner her en node mens nodenes topologipakker er på vei rundt ringen. Det som da kan skje er at man kan miste pakker på veien eller at informasjonen kommer tilbake til avsender uten å nødvendigvis være rett. Informasjonen kan være feil fordi noden kan ha blitt fjernet straks etter å ha lagt til sine MAC-bindinger i en pakke. Dette avhenger også av hvilken metode for nettverkskartlegging man bruker. Informasjonen vil da bli ført tilbake til avsender der den prøver å lage et topologibilde. Det er da er fare for at innholdet i topologibildet ikke blir rett. I hendelser som denne over som det er mindre sannsynlig at vil skje på RPR-ringen kan man tillate en lengre tid for oppdatering enn hendelser som skjer oftere.

La oss si det skal legges til en node i RPR-ringen. I dette tilfelle vil det typisk være flere situasjoner som oppstår som nettverkskartlegging må kunne håndtere. En kan tenke seg at det oppstår et brudd når operatøren kobler den nye noden til RPR-ringen. Nå må mekanismen oppdage den nye topologien til nettet og samarbeide med andre mekanismer for å eventuelt gjennomføre wrapping og styring. Når så denne nye noden er klar for å sende pakker på ringen vil nettet igjen få en ny topologi som mekanismen må oppdage. Testene som gjennomføres med simulatoren gjennomgår mange situasjoner som oppstår ved reelle hendelser på ringen men tester de enkeltvis. I tilfellet over vil det være to uavhengige tester som kjøres på ringen.

De testene som kjøres skal samlet kunne dekke over mange av hendelsene som kan oppstå på RPR-ringen.

6.1.4 Nødvendige endringer i simulatormodellen

For at simulatormodellen skulle kunne fungere til alle testene måtte det gjøres relativt store endringer. Hovedgrunnen til dette var det faste oppsettet med 16 noder. Det som gjorde endringer vanskelig var at det ikke var enkelt å endre topologien mens simulatoren kjørte. Det var satt opp en tabell med det antall noder som simulatormodellen skulle starte med. Tillegging av en node som skulle inn i tabellen var derfor ikke enkelt fordi det ikke var plass i tabellen til flere noder. Fjerning av noder krevde også endel endringer i koden på grunn av applikasjoner og flytkontroll som kjørte på nodene. Simulatoren måtte også gjennom endel endringer med tanke på håndtering av feil. For å teste nettverkskartlegging ville jeg blant annet påføre linkene feil og ta bort noder. Alle nodene sendte blant annet data til hverandre uten noen sjekk om mottakeren de prøvde å sende til faktisk var på ringen.

6.1.5 Oppsett av simulatormodellen

Simuleringene er basert på en RPR-ring med 16 noder. Min egen mekanisme er også testet med maksimalt antall noder på RPR-ringen (255). Resultatene fra min egen metode vil følge i et senere kapittel. Linkene har lik lengde på 2km noe som gir en propagasjonstid på 10μs. Det regnes med en hastighet på 2*10$^8 \frac{m}{s}$ (hastigheten på data over linkene). Det vil da følgelig ta lengre tid for topologipakkene å gå rundt ringen. Ved testene der det blir lagt til en eller flere noder vil det etter å ha blitt lagt til noder være 16 (evt. 255) tilsammen på RPR-ringen. Tillegging av en node gjøres ved å først fjerne en node og la nettverkskartlegging stabilisere seg i nodene. Hva som er stabil tilstand varierer for de ulike metodene og må vurderes for hver av metodene og hver av testene. Deretter vil en node bli lagt til for å teste hvordan mekanismen håndterer dette. 16 er valgt av flere grunner. Da jeg begynte å videreutvikle simulatormodellen var denne satt opp med 16 noder i en fast topologi. Det var derfor naturlig å fortsette med denne topologien på 16 noder. Det er interessant å se hvordan nettverkskartlegging fungerer også med maksimalt antall noder for å verifisere at den også kan vedlikeholde topologibildene ved et slikt oppsett.

Simulatormodellen tar ikke hensyn til tid som brukes til prosessering av de forskjellige mekanismene. Det som bruker tid i simulatoren er flytting av pakker langs en link, legge pakker inn i buffere, ta pakker ut av buffere og vente hvis det er to eller flere pakker som skal inn i ett buffer på en gang.

6.1.6 Usikkerheter i simulatormodellene

Selv om noe av programkoden var gjenbruk var programmeringen av simulatormodellene en omstendig prosess og en vesentlig del av arbeidet med denne oppgaven. Hver av simulatormodellene er på circa 4000 linjer kode. På grunn av størrelsen på disse programmene er det mulighet for at det finnes feil i kildekoden. Siden resultatene i mange tilfeller er vanskelig å forutse er det vanskelig å vite om det er feil i koden hvis man får et resultat man ikke hadde ventet.

Utkastet er, som ordet sier, enda bare et utkast, noe som gjør at innholdet av utkastet er mangelfullt og kan også være feil. Noen av manglene er deler av mekanismen som trengs ved utvikling av simulatormodellen. Slike punkt er tatt opp som problemer under implementering da de som regel har kommet opp underveis.

6.2 Darwin

Dette delkapittelet tar for seg Darwin-forslaget. Det er satt opp et delkapittel som tar for seg hvordan mekansimen er antatt å fungere. De faktiske resultatene blir så presentert før resultatene drøftes.

6.2.1 Hvordan mekanismen er antatt å fungere

Jeg vil i denne delen drøfte hvordan mekanismen er antatt å fungere. Dette vil bli bekreftet eller avkreftet under simuleringen og drøftingen av resultatene senere i oppgaven.

Denne metoden benytter seg ikke av noen koordinert mekanisme for nettverkskartlegging. Dette fører til at hver enkelt node på RPR-ringen selv må finne ut nettets topologi uten å få noe hjelp av de andre nodene. Det betyr at hvis en av nodene finner en endring i nettets topologi har den ingen mulighet for å fortelle de andre nodene om det som har skjedd. Dette er et punkt som vil kunne ha negativ innflytelse på ytelsen til mekanismen. Hvis ikke beskyttelsemeldinger, i noen tilfeller, vil utløse nettverkskartlegging i alle nodene vil det være mulighet for at noen av nodene må vente så lenge som perioden for regelmessig nettverkskartlegging før de får oppdatert sitt topologibilde. Dette vil sannsynlig gjelde for alle endringer av nettets topologi. Ved tillegging av en node vil noden som legges til starte en oppdatering av sitt eget topologibilde, men har ingen mulighet til å få de andre nodene til å starte en oppdatering av deres topologibilde.

6.2.2 Gjennomføring

Testene er vist på side 35. Simuleringene på Darwin-forslaget er basert på at det må komme to pakker som gir et topologibilde annerledes enn det topologibildet som noden allerede har før det lokale blir endret. Dette er nevnt tidligere i kapittel 5.3.5.

6.2.2.1 Oppbygging av topologibildet For at mekanismen “ringlet selection” skal bli så effektiv som mulig er det viktig at topologibildet bygges opp på en slik måte at det er effektivt å søke i datastrukturen.

I min simulator vil jeg utvide funksjonaliteten til en trestruktur som finnes i Java biblioteket. Strukturen kalles en “TreeMap” og implementerer et Rød-Svart tre. Navnet til strukturen kommer av måten et slikt tre implementeres på der nodene i treet blir tilordnet status som “Rød” eller “Svart”. Røde og svarte noder relaterer til hverandre etter fastsatte regler. Disse reglene vil ikke bli nevnt her. Implementasjonen av et Rød-Svart tre gjør at man, med liten tidskompleksitet, finner en node i treet. Men N noder i treet garanterer treet en tidskompleksitet på log(N) når en verdi i treet skal finnes. Nodene i treet har en referanse til innholdet i den aktuelle noden. Treet blir i simulatormodellen bygget opp med nodenes identifikator

på RPR-ringen som noder i det Rød-Svarte treet. Hver node i treet inneholder antall hopp til mottakeren og hvilken ring mottakeren kan nås på.

6.2.3 Resultat

Flere av resultatene har en beskrivelse som sier “etter andre runde nettverkskartlegging”. De forskjellige nodene behøver ikke å sende sine topologipakker samtidig for at mekanismen skal fungere. I Darwin er hver enkelt nodes mekanisme for nettverkskartlegging uavhengig av de andres. “Andre runde” vil da bety at alle nodene på RPR-ringen har sendt og mottatt sine topologipakker to ganger. Hvor lang denne tiden er avhenger av to faktorer. For det første timerperioden, men også tiden fra endringen skjer til alle nodene har sendt ut topologipakker. Det er også mulig at en node som allerede har sendt ut topologipakker når endringen skjer kan merke at det har skjedd en endring. Topologipakkene oppdager endringen som har skjedd på ringen fordi endringen skjer før pakkene når punket på ringen der det har skjedd en endring. Tabell 3 viser resultatene fra simuleringer av Darwins forslag.

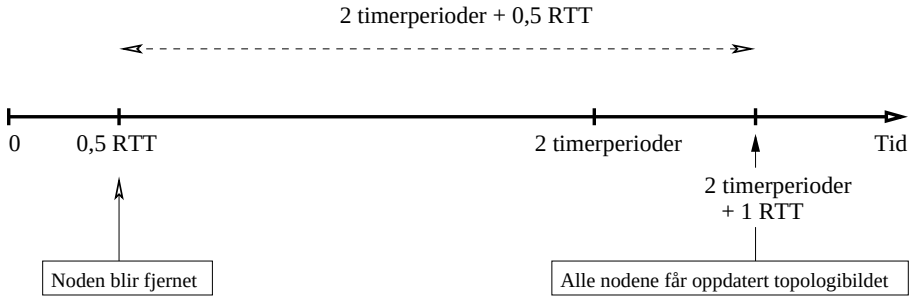
Test nr.	**Resultat**	**Kort kommentar**
1	1 RTT	
2	To timerperioder for nettverkskartlegging + 0,5 RTT.	Dette er illustrert i figur 24 på neste side.
3	Etter andre runde nettverkskartlegging etter noden ble fjernet.	Lengste mulige periode blir da to timer perioder pluss en rundetid.
4	Etter andre runde nettverkskartlegging etter noden ble lagt til.	Lengste mulige periode blir da to timer perioder pluss en rundetid.
5	Etter andre runde nettverkskartlegging etter noden ble endret.	Lengste mulige periode blir da to timer perioder pluss en rundetid.
6	Etter andre runde nettverkskartlegging etter linken ble brutt.	Tiden topologi pakkene bruker rundt ringen har blitt lengre.
7	Nodene i begge deler av det isolerte nettet finner ny topologi etter andre runde nettverkskartlegging.	
8	Nodene setter nettet sammen igjen etter to runder.	

Tabell 3: Tabellen viser resultatet fra testene kjørt med Darwins forslag.

6.2.4 Drøfting

6.2.4.1 Oppstart - test 1 I test nummer 1 sender alle nodene ut sine to pakker samtidig. Når propagasjonstiden er lik for alle linkene kommer det to pakker inn i hver node samtidig

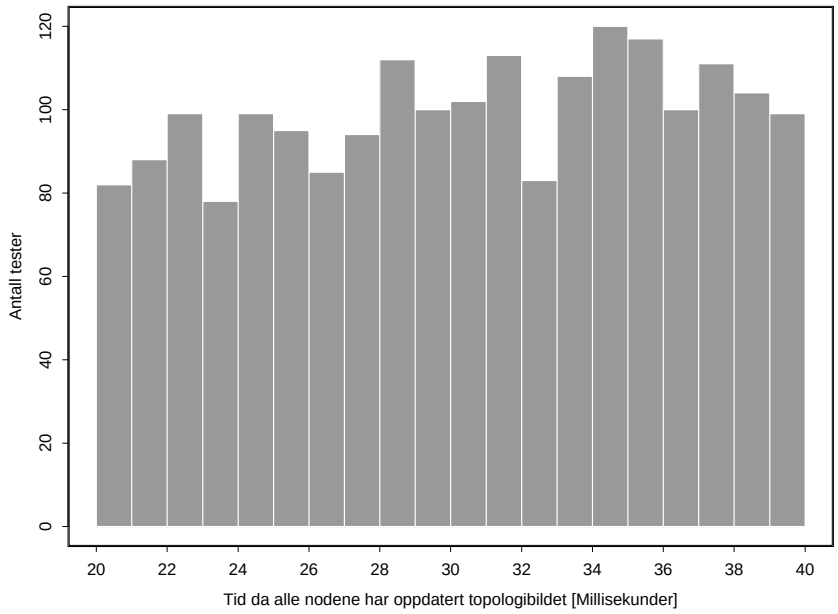
som skal prosesseres. Prosesseringstid er ikke modellert i simulatoren. Derfor vil alle nodene få oppdatert topologibilde innen 160 μs (1 rundetid). I en virkelig RPR-ring må man også regne med at prosessering i hver node kan ta litt tid. Det er derimot ikke en veldig kompleks prosess. En rundetid vil på grunn av prosessering derfor være mer enn bare linkenes samlede propagasjonstid.



Figur 24: Figuren viser hvorfor tiden fra noden blir fjernet til oppdatering blir 2 timerperioder + 0,5 RTT i test nummer 2.

6.2.4.2 Fjernet node ved oppstart - test 2 I test nummer 2 ble det fjernet en node fra ringen etter 80 μs. Dette er halvparten av tiden målt i test nummer 1 og også en halv RTT. Resultatet av denne testen avhenger av hvor pakkene er på linken akkurat da noden feiler. I en av mine tester fjernet jeg en node akkurat da denne noden hadde mottatt to topologipakker. På grunn av at tiden var akkurat halvparten av propagasjonstiden var begge pakkene som var i noden fra samme node på RPR-ringen. Dette førte til at avsenderen av disse pakkene ikke fikk noe topologibilde denne første runden. I de andre nodene varierte det om de hadde 14 eller 15 noder i sitt topologibilde avhengig av hvilken av de to pakkene som blir først prosessert i avsendernoden. Dette kommer av at de to pakkene blir prosessert separat. Noen av nodenes topologipakker har allerede passert noden på ringen som fjernes. Disse vil inneholde 15 forskjellige noder, mens topologipakkene som ikke har passert noden som fjernes vil inneholde 14 noder når de kommer tilbake til avsenderen. Slik topologibildet ble bygget her har ikke nodene seg selv i topologibildet. I de tilfellene der nodene har 15 andre noder i sitt topologibilde er noden nødt til å vente to timerperioder på grunn av at det må komme to pakker som verifiserer den “nye” topologien. Den ene noden som ikke fikk noe topologibilde i den første runden får derimot sitt oppdatert og riktig den andre runden. Denne må ikke vente to runder på grunn av at når andre runde starter vil den ikke ha noe topologibilde. Varierer man tiden for fjerning av en node under en allerede startet runde nettverkskartlegging vil antall andre noder i RPR-ringens topologibilder variere mellom 14 og 15. De som har 15 noder i topologibildet har da, etter fjerning, en node for mye i sitt topologibilde, mens de som har 14 noder i topologibildet har rett bilde. Ut fra disse resultatene kan man konkludere med at tap av topologipakker er uheldig fordi oppdatering av topologibildet i nodene blir forsinket. Tapte topologipakker vil derimot ikke påvirke senere pakker som blir sendt og disse kan da oppdatere topologibildet.

6.2.4.3 Fjerning av node - test 3 Ved fjerning av en node fra ringen vil dette bli oppdaget ved neste runde nettverkskartlegging, og topologibildene endret i runden etter dette. Tidspunktet for når den første runden etter fjerning blir gjennomført vil bli bestemt av perioden for nettverkskartlegging eller om beskyttelsemekanismen utløser utsending av topologipakker. Hvis fjerningen av noden er av en slik karakter at noden selv får sendt ut beskyttelsemeldinger som utløser en runde nettverkskartlegging vil den første runden starte umiddelbart. Hvis ikke vil man måtte vente til neste runde regelmessig utsending av topologipakker. Figur 25 viser fordelingen av resultat ved denne testen. Resultatene er relativt jevn fordelt mellom en og to timerperioder.



Figur 25: Bildet viser fordelingen av resultat ved fjerning av en node med Darwins metode. Det er 16 noder på RPR-ringen. Legg merke til at x-aksen starter på 20ms.

6.2.4.4 Fortsettelse test 3, lagt til node - test 4 og endring av identifikator - test 5 Resultatet i testene 3, 4 og 5 kan generaliseres. Fjerning av en node, tillegging av en node og endring av identifikator på en node gir samme resultat. Brukes nettverkskartlegging alene må nodene vente opptil en timerperiode før en endring i topologien blir oppdaget av mekanismen, og eventuelt enda en runde før topologibildet blir endret.

6.2.4.5 Bryting av en link - test 6 Darwin har opsjoner i topologipakkene som kan fortelle om noden har støtte for wrapping og om noden er i en wrappet tilstand. Dette siste feltet har til hensikt å fortelle de andre nodene på ringen at det er en feil. De to nodene på hver side av feilen setter dette bitet i sin MAC-tilleggsinformasjon i topologipakkene før de sender de videre. Dette gir nodene nok informasjon til å gjennomføre styring hvis dette er ønskelig.

Pakkene går da i en slik retning at de ikke går gjennom en node som er wrappet. Simulatormodellen ble implementert slik at hvis nettverkskartlegging oppdaget at ringen var wrappet, gjennomførte den styring da topologibildet skulle bygges opp. Igjen var det slik at det måtte to runder til for at topologibildet skulle bli oppdatert, og i mellomtiden ble pakker på RPR-ringen wrappet. Når topologibildet blir oppdatert tar mekanismen til bruk "wrapped" feltet for å vite når det er mere optimalt å sende pakker den andre veien enn ved normal operasjon på ringen. Ved hjelp av informasjonen i MAC-bindingene bygger nodene opp et topologibilde der den korteste veien til de andre nodene på ringen finnes. Topologibildet vil, med algoritmen under, inneholde den avstanden til de andre nodene der data ikke blir wrappet. Algoritmen for å utarbeide den mest optimale veien er som følger:

```
// Tabell av MAC–bindinger fra mottatt topologipakke
mb = pakke.bindinger;

// Ringen der topologipakken ble mottatt
dir = direction;

// ringwrapped vil indikere om noen tidligere prosessert node er wrappet
ringwrapped = node.wrappet;

teller = 1;

while ( <mb har flere poster> )
{
    if (ringWrapped)
        {
            dir = <motsatt ring enn den der pakken ble mottatt>;
            len = <totalt antall bindinger> – teller;
        }
      else
        len = teller;

      if ( <noden ikke i topologibildet> )
      {
          <legg til noden>
      }
      else if ( IKKE ringwrapped )
      {
          if ( <funnet avstand er kortere enn lagret> )
              <overskriv lagret verdi>;
      }
      else
      {
        <legg til noden>
      }

      teller++;

}
```

Darwins nettverkskartlegging finner den nye topologien ved wrapping uten problemer. Det som skiller mekanismen ved wrapping og ved normal operasjon er tiden topologipakkene bruker på å komme tilbake til seg selv. Uten wrapping er det 1 rundetid. Med én brutt bidireksjonal link på RPR-ringen vil pakkene gå over 2*(N-1) linker, der N er antall noder i nettet.

6.2.4.6 Bryting av to linker - test 7 I test 7 simuleres feil på to linker slik at det blir dannet to separate nett. To sett av noder blir isolert fra hverandre der det ene settet kan være så lite som én node. Uansett størrelse på de to separate nettene fungerer mekanismen feilfritt. Algoritmen over tar hånd om nettverkskartlegging for hver node og bygger opp det lokale topologibildet med de nodene som kan nås.

6.2.4.7 Sammensetting av brutt nett - test 8 I test 8 settes de to isolerte delene sammen igjen. Darwins topologipakker fanger opp endringen etter den første gangen topologitimeren går ut men topologibildene blir ikke endret før andre runden.

6.3 Utkast 0.3 / 1.0

Denne delen tar for seg forslaget til nettverkskartlegging som kommer frem i utkast 0.3 og utkast 1.0. I testingen av utkastets forslag vil jeg, der det er hensiktsmessig, ta for meg forbedringer ved å bruke forslaget i utkast 1.0 i forhold til 0.3. Oppbygningen av kapittelet er den samme som for testingen av Darwin. Først antagelser om hvordan mekanismen er fungerer. Deretter kommer gjennomføring med resultater og drøfting.

6.3.1 Hvordan mekanismen er antatt å fungere

Utkastets forslag oppfører seg på en annen måte enn Darwin. Utkastets mekanisme sender topologipakker ikke bare basert på timere, men også basert på pakker som kommer fra de andre nodene på ringen. Koordinert nettverkskartlegging er tidligere diskutert i kapittel 5.3.3 på side 28.

Siden denne metoden benytter en koordinert metode for nettverkskartlegging kan det være hensiktsmessig å tro at den presterer bedre enn Darwins forslag. På grunn av koordineringen vil de forskjellige nodene opplyse hverandre om en forandring i RPR-ringens topologi. Denne koordinerte mekanismen i forslaget er derimot ikke fullstendig slik at koordineringen kan fungere i noen situasjoner og ikke fungere i andre.

Hvis en node blir fjernet er det bare nabonodene som sender ut nye topologipakker. Det er riktig at det kun er disse to som har fått nye naboer, men antall hopp og retning til noen noder kan også endre seg i topologibildene i noen av nodene på RPR-ringen. Dette kan føre til at topologibildet til alle nodene ikke blir oppdatert før alle noder har sendt ut og mottatt sine topologipakker. De andre nodene må også fjerne en node fra det topologibildet de har.

Ved endring av identifikator på en node vil de andre få oppdatert topologibildet når noden som endrer sender ut topologipakker.

Utkast 0.3s motstridende informasjonen om utløsing av sending av topologipakker er nevnt i underkapittel 5.3.6.3 på side 32 i denne oppgaven. Det kan skje en endring i nettets topologi som ikke utløser utsending av topologipakker med nabonodene satt til 0. Hvis to

noder da merker at de har fått en ny nabo vil de sende ut topologipakker. Hvis ikke de andre nodene sender ut topologipakker når de merker en endring i nettets topologi vil dette kunne føre til at enkelte avstander i noen topologibilder blir feil. Simulatoren bør derfor lages med mulighet for å prøve ut at bare nabonodene sender ut pakker slik at dette kan bli bekreftet eller avkreftet.

I utkast 1.0 derimot står det at nodene skal sende ut topologipakker på alle linker i alle situasjoner der de oppdager en endring i nettets topologi eller endrer sitt eget topologibilde.

Metoden er koordinert ved at nye noder utløser en oppdatering av topologibildet i de andre nodene. Alle nodene sender ut nye topologipakker når de mottar topologipakker fra andre der begge nabonodene er satt til null. Dette indikerer en ny node på RPR-ringen som trenger informasjon om nodene som allerede finnes på RPR-ringen. De andre nodene får derimot ikke informasjon om den nye nodens naboer i denne første meldingen. Det vil da ta litt tid for at alle nodene får oppdatert informasjon om den nye nodens naboer inn i topologibildet.

Denne koordinerte mekanismen kan også føre til at mange topologipakker blir sendt ut på RPR-ringen samtidig fra de forskjellige nodene. Hver gang en node utløser utsending av topologipakker basert på andre noders topologipakker blir det sendt ut en pakke på hver ring. Dette er ikke en stor mengde pakker, men det som kan skje er at en node mottar mange slike utløsere på kort tid og dermed genererer mange topologipakker. Dette kan føre til at disse pakkene blir forsinket i forhold til ideell propagasjonstid uten forsinkelse.

6.3.2 Gjennomføring

Fordi simulatoren også skulle kunne brukes for å teste denne metoden for nettverkskartlegging måtte det lages en helt ny kode basert på utkastene. Simulatormodellen ble også skrevet om litt i 'Node' klassen som tar seg av det som skjer inne i en node. I hovedsak var det metoden "outputSelector" som ble endret. Denne har som oppgave å velge fra hvilket buffer i hver node neste pakke skal sendes fra. Metoden ble skrevet om for å ligne mest mulig på måten det blir gjort i utkastet. Et tilstandsdiagram fra utkast 0.3 ble implementert. Diagrammet kan finnes i forenklet versjon i Appendix A på side 83.

For å teste mekanismen for nettverkskartlegging kreves strengt tatt ingen datatrafikk på ringen. Hvis nettets topologi er ukjent for nodene vil ingen datapakker slippes ut på ringen. Ser man bort fra lavprioritets datatrafikk som reguleres av fairnessalgoritmen er rekkefølgen pakker skal velges fast. Det er også en mekanisme som sørger for at ikke bufferene går fulle slik at pakker tapes. Rekkefølgen av pakker som da velges kan skille seg ut fra normalen. Så lenge ikke bufferenes datamengde nærmer seg bufferenes kapasitet vil ikke lav- eller høyprioritets datatrafikk påvirke nettverkskartlegging merkbart. Bufferene kan nærme seg fulle hvis det er veldig høyt påtrykk av data på RPR-ringen. Topologipakker har høy prioritet på RPR-ringen. Den eneste måten man kan merke utslag på nettverkskartlegging er en minimal forsinkelse ved veldig høyt påtrykk av høyprioritets datatrafikk. Utkastet sier derimot at høyprioritets datatrafikk har minimal forsinkelse og jitter. Det finnes også mekanismer på RPR-ringen for å reservere båndbredde hvis dette blir aktuelt for nettverkskartlegging. På grunn av dette er det i denne simuleringen ikke tatt med lavprioritets datatrafikk. Dette er også på grunn av fairnessalgoritmen som ikke er relevant for problemstillingen i denne oppgaven.

6.3.3 Problemer og mangler under implementering

Under implementeringen av de ulike mekanismene er det noen problemer og mangler som viser seg. Dette er ting som må tas hensyn til under implementeringen, men som utkastet ikke gir noen løsning på. To slike eksempler vil bli beskrevet under.

6.3.3.1 Wrapping Wrapping er en mekanisme som er mangelfullt forklart i utkast 0.3. Utkast 1.0 løser noen av disse problemene. Utkast 1.0 sier at topologipakker ikke skal wrappes. Hvis man da wrapper en link på RPR-ringen når alle nodene er i en stabil tilstand med et fullt topologibilde må man avgjøre om det nye topologibildet skal bygges over det som ikke lenger er gyldig, eller om det gamle topologibildet skal slettes fullstendig før nytt topologibilde blir bygget opp av nodene. Hvis man legger til grunn at man tar vare på postene i topologibildet vil den nye faktiske avstanden til noen av nodene i topologibildet endre seg. Avstandene vil kunne bli lengre, og nodene må da vite at dette er den eneste måten å nå noden på og derfor bytte ut posten i topologibildet som gir veien til den aktuelle noden. Hvordan noden skal vite om hvilke poster i bildet som har blitt ugyldig må bli gitt av mekanismen for nettverkskartlegging. Hver node kan kun nås på én av ringene fra en vilkårlig avsendernode. Dette fører også til at noen poster må slettes fra topologibildet siden utkast 1.0 tar vare på avstanden til alle nodene på begge ringene i hvert sitt topologibilde. Sletting av nodenes topologibilde hvis wrapping blir satt i gang vil gjøre at nodene kun baserer gjenoppbyggingen av topologibildet på de nye pakkene som kommer inn langs hver av ringene. Slik vil de automatisk bli kvitt de gamle postene som ikke lenger er riktige og få oppdatert de nye. Det krever derimot at nodene er klar over at det har skjedd wrapping på RPR-ringen og dermed sletter sitt topologibilde.

Som et forslag på løsning av problemet nevnt over kan beskyttelsemeldinger brukes. Disse rapporterer til nodene hvis det skjer noe med en link på ringen. Disse meldingene kan også brukes av mekanismen for nettverkskartlegging for at denne skal fungere i situasjonen som er nevnt i avsnittet over. Når beskyttelsemeldingene indikerer en endring i nettets topologi der nodene skal bygge opp en nytt topologibilde kan de slette topologibildet de har lagret for å så bygge opp et nytt. Det er også mulig å slette et subsett av de lagrede postene for at topologibildet skal holde seg konsistent selv etter wrapping er satt i gang. Når en node mottar en beskyttelsemelding fra en annen node om brutt link vil noden slette alle de postene som er lengre fra avsendernoden i sine topologibilder (ett for hver ring). Utkast 1.0 fremmer et slikt forslag der beskyttelsemeldinger gjør endringer på topologibildet. Jeg vil i min simulator følge utkast 1.0, men også teste situasjonen der alle postene i topologibildet slettes. Dette er derimot et avvik fra selve utkastet, men vil være en annen mulig løsning på et problem med wrapping. Nodene trenger da, før de mottar ny topologiinformasjon, en melding som sier at de skal slette sitt topologibilde. Dette kunne vært en del av topologipakkene, men et slikt forslag er ikke lagt frem. Når de har slettet sitt topologibilde har ikke nodene mulighet til å bruke RPR-ringen før de har fått et nytt topologibilde.

6.3.3.2 Konsistenssjekk av topologibilde For å sjekke når nodene når en konsistent / komplett tilstand brukes en konsistenssjekk. Tiden mekanismen bruker for å oppdatere topologibildet etter testene varierer, og topologibildet vil gradvis bygges opp etter feilen/endringen er påført. Ettersom hvilke av testene som kjøres vil topologibildet variere i hvordan det skal se ut. Konsistenssjekken vil da kontrollere om topologibildene i de forskjellige nodene ser ut

som de skal. For eksempel ved testing av brudd på en link vil de to topologibildene, ett for hver ring, inneholde informasjon som ikke kan sjekkes opp imot hverandre. Nodene vil da bare ha informasjon på hver ring om de nodene som faktisk kan nås langs denne ringen uten å wrappe pakker. Bortsett fra noden selv som kan "nås" med 0 hopp vil settet av noder som finnes i hvert av de to topologibildene være disjunkte. Antall noder tilsammen i de to topologibildene vil derimot kunne sjekkes opp i mot det totale antallet noder som finnes i den brutte RPR-ringen. Dette er informasjon som nodene ikke vet sikkert, men informasjonen finnes i simulatoren og kan brukes til en sjekk av topologibildet. Dette vil være en test som nodenes mekanisme for nettverkskartlegging ikke har mulighet til å kjøre på grunn av at den bruker informasjon som finnes i alle nodene. Nodene på RPR-ringen har bare tilgang til informasjon som finnes i sitt eget topologibilde og informasjon som distribueres under selve mekanismen for nettverkskartlegging. Simulatoren har derimot tilgang til alle nodenes topologibilde og annen statusinformasjon som finnes i simulatoren.

6.3.4 Resultat

De testene som gir ulik tid for oppdatering for hver test er kjørt flere ganger. Flere av testene er kjørt 2000 ganger og gjennomsnittet av disse er presentert. Tallet 2000 er valgt for at resultatene som kommer fra testene skal gjenspeile den virkelige trenden i fordelingen av resultatene. Mange av testene som er presentert med fordeling på 2000 tester ble først vurdert etter 1000 tester. 1000 nye tester ble deretter kjørt for å se om trenden på fordelingen av resultatene endret seg. Endringer på resultatene fra 1000 til 2000 var minimale, noe som indikerer at 2000 gjenspeiler trenden i den virkelige fordelingen. Disse verdiene og fremgangsmåten for å teste de er ikke basert på statistiske bevis.

Resultatet av testene er presentert i tabell 4 på neste side. I underkapittel 6.3.5 blir resultatene gått nøyere igjennom og drøftet.

6.3.5 Drøfting

Kun én pakke kan legges ut på linken på en gang. Hvis det er flere pakker som kjemper om å bli sendt må noen vente på tur. Denne ventingen gir utslag i noen av resultatene. Tidene som dette utgjør er derimot små i forhold til de totale tidene målt.

6.3.5.1 Oppstart - test 1 Ved oppstart har alle nodene blankt topologibilde. Dette fører til at de må sende ut topologipakker der nabonodene er satt til 0. Dette utløser en ny sending av statusmeldinger fra naboene. Hvis linklengden på to nabolinker er lik vil man kunne komme i en situasjon der flere pakker skal sendes samtidig. En nodes nabolinker betegner da linkene ut på hver side av en node som strekker til hver av nodens naboer langs RPR-ringen.

Ved kjøring av test 1 på dette forslaget gir det et resultat på 1 rundetid for at alle nodene får et komplett topologibilde av ringen. Da går vi fra en tilstand der ingen av nodene har noe topologibilde fra før. I tillegg til rundetiden kan det, avhengig av linklengdene, bli litt ekstra forsinkelse. Denne problemstillingen ble tatt opp i forrige avsnitt.

6.3.5.2 Fjernet node ved oppstart - test 2 Test nummer 2 viser at ved fjerning av en node midt i kartleggingsprosessen vil et komplett topologibilde ikke oppnås før neste runde

Test nr.	**Resultat**	**Kort kommentar**
1	1 RTT (160μs)	
2	En timerperiode for nettverkskartlegging.	
3	10048 μs. ($\approx$0,5 timerperiode)	Gjennomsnittstid etter 2000 kjørte tester. Den totale fordelingen er vist i figur 26.
4	240μs. (1,5 RTT)	
5	10270μs. / 160μs. ($\approx$0,5 timerperiode / 1 RTT)	Henholdsvis (ingen sending) / (sending) av topologipakker ved endring av identifikator.
6	10090μs. / 300μs. ($\approx$0,5 timerperiode / $\approx$2 RTT)	Henholdsvis (ingen sending) / (sending) av topologipakker ved bryting av link.
7	9785μs. / 70μs. ($\approx$0,5 timerperiode / $\approx$0,4 RTT)	Henholdsvis (ingen sending) / (sending) av topologipakker ved bryting av link. 0,4 RTT kommer av at det har blitt to nett. 1 RTT er forsatt rundetiden på den feilfrie ringen.
8	10121μs. / 190μs. ($\approx$0,5 timerperiode / $\approx$1,2 RTT)	Henholdsvis (ingen sending) / (sending) av topologipakker ved gjenoppretting av linkene.

Tabell 4: Tabellen viser resultatet fra testene kjørt med forslaget i utkast 0.3 / utkast 1.0. Resultatene uten pakkesending er gjennomsnittsverdier. Det er 16 noder på RPR-ringen og linkene mellom hver node er 2km lange (10μs).

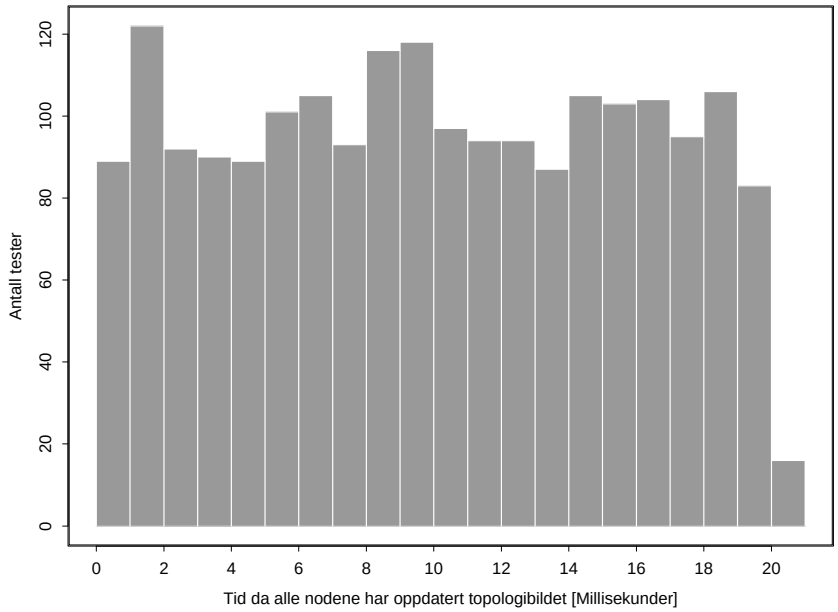
nettverkskartlegging kjøres. Neste runde er litt uklart begrep med tanke på at alle nodene behøver ikke igangsette nettverkskartlegging samtidig. Hvis alle nodene bruker samme tid for periodisk nettverkskartlegging vil alle nodene ha sendt sine topologipakker innen denne perioden. Topologibildet blir konsistent når en node har mottatt og prosessert topologipakker fra alle de andre nodene på RPR-ringen. Ved å sjekke sitt eget topologibilde for konsistens vil nodene kunne ha en formening om deres eget bilde gjenspeiler nettets virkelige topologi. Topologibildet blir inkonsistent ved oppstart på grunn av at så snart hver node blir startet sender de ut informasjon om seg selv til de andre nodene. Om noden da går ned under eller etter runden med nettverkskartlegging er ferdig har ingenting å si. Dette kommer av at alle de andre nodene vil legge til den fjernede noden i sitt topologibilde selv om noden ikke finnes i der. Når en node mottar en topologipakke må de regne med at avsenderen finnes på RPR-ringen. Pakkene som noden sender ut ved oppstart vil derimot ikke bli fjernet fra ringen av noden selv fordi den er borte, men må sirkulere til TTL blir 0.

6.3.5.3 Fjerning av node - test 3 Det samme kommer frem i test 3 som i test 2. Fjerning av en node blir oppdatert hos alle nodene når alle nodene har sendt rundt ringen sine topologipakker neste gang. Alle nodene på RPR-ringen vil måtte vente på pakker fra alle de andre nodene før får oppdatert sitt topologibilde. De må også ha en klar formening om hvordan de skal hindre den siste posten i topologibildet for å være gyldig. Dette er forklart i delkapittel 5.3.6.1 på side 30. I simulatoren bruker nodene det antallet noder de mener finnes på ringen til å hindre bruk av den siste posten i topologibildet.

Det vil ikke være noen mekanisme på ringen som vil sørge for at nodene sier ifra til hverandre hvis en node har blitt fjernet. For at topologibildene i alle nodene da skal bli konsistent må alle nodene for seg selv ha gjennomført en runde nettverkskartlegging. Nodene vil da få all informasjon om naboer og avstand til de enkelte nodene fra pakkene som kommer fra de andre, og funnet ut hvor mange noder det er på ringen ut fra sine egne pakker. Hver nodes pakkesending i dette tilfelle vil være uavhengig av hverandre. Den gjennomsnittlige tiden det vil ta for at alle nodene har oppdatert sin topologibilde vil da være lik den gjennomsnittlige tiden det tar før en sender ut pakker pluss ringens rundetid. Tiden for periodisk nettverkskartlegging er i dette forslaget satt til å være 20 millisekunder. Det ble kjørt 2000 målinger på denne testen der snittverdien på tid før alle har nytt topologibilde ble 10048 μs. Resultatene er relativt jevnt fordelt mellom 0 og 20 millisekunder. Dette er veldig nærme snittid for forventet pakkesending siden denne vil være 10 millisekunder. 10 millisekunder kommer av at ved et vilkårlig tidspunkt vil tiden det tar før en node sender ut topologipakker ligge mellom 0 og tiden for periodisk nettverkskartlegging på circa 20ms. Tidspunktet for fjerning av noden vil være vilkårlig og de 2000 kjøringene vil ha ha tider som er uavhengig av hverandre. Figur 26 på neste side viser fordelingen av de 2000 målepunktene.

6.3.5.4 Lagt til node - test 4 Når en node, som i test 4, blir lagt til på ringen vil den selv sette i gang en runde nettverkskartlegging. Den vil da sende ut nye pakker med sine naboer satt i pakkene til å være 0. Dette vil utløse pakkesending i alle nodene slik at den nye på RPR-ringen vil kunne oppdage hvilke andre noder som finnes der. Det vil også få de andre nodene til å oppdatere sine topologibilder. Det vil være andre avstander mellom de ulike etter det har blitt lagt til én eller flere noder. Tiden det tar før alle har oppdatert sitt topologibilde vil avhenge av rundetiden i RPR-ringen. Pakkene den nye noden sender ut vil nå nodene på RPR-ringen etter tur og den siste noden pakkene kommer til er den siste noden som oppdager en endring. Denne vil så sette i gang en runde nettverkskartlegging selv. Pakkene denne sender må gå rundt RPR-ringen til de kommer tilbake til avsenderen før topologibildet har blitt oppdatert. Dette vil samlet gi en oppdateringstid på circa 1,5 rundetid. Det vil ta en halv rundetid å nå noden lengst fra den nye noden og deretter én rundetid for at pakkene denne noden sender skal gå rundt RPR-ringen 1 gang.

6.3.5.5 Endring av identifikator - test 5 RPR bruker MAC-adresser som identifikator på nodene i RPR-ringen. MAC-adresser følger nettverkskort men de kan overstyres av en administrator hvis dette er interessant. I dette spesialtilfellet av hendelser på RPR-ringen vil dette være en situasjon der nodens identifikator på RPR-ringen vil endre seg. Dette krever at topologibildene langs RPR-ringen må oppdateres med den nye MAC-adressen for den spesifikke noden. Det er ikke nevnt noe om hvordan dette skal gjennomføres i RPR med tanke på nett-



Figur 26: Bildet viser fordelingen av resultat ved fjerning av en node med utkastets metode. Det er 16 noder på RPR-ringen.

verkskartlegging. Noden kan selv sende ut topologipakker etter å ha endret identifikator. Dette vil føre til at alle de andre på ringen vil få oppdatert den rette posten i sitt topologibilde fordi den lagrede identifikatoren er en annen enn den som mottas, mens antall hopp er det samme. Den gamle vil da skrives over av den nye. Hvis derimot noden ikke sender ut oppdatering via topologipakker etter å ha endret identifikator vil en bli nødt til å vente på at timeren skal gå ut før ny identifikator blir propagert rundt ringen. Testene som er gjennomført med endring av identifikator på en node stemmer også med hvordan mekanismen er antatt å fungere. Tiden det tar å oppdatere topologibildene til alle nodene på ringen er gjennomsnittiden for at en node sender ut topologipakker i tillegg til én rundetid. Dette resultatet er basert på at ingen topologipakker blir sendt ut som følge av endringen av identifikator, men kun som følge av at timerene utløser når de når 0. Det målte resultatet er 10270μs ($\approx$0,5 timerperiode). Denne tiden kan derimot forbedres radikalt hvis nodene "sier ifra" til de andre at den har endret identifikator. Endringer som må gjøres i de andre nodene er at 1 post må endres til å ha den nye identifikatoren. I tillegg må alle poster som inneholder den endrede som nabonode oppdateres. Etter en veldig enkel endring i simulatormodellen startet mekanismen en runde nettverkskartlegging ved endring av identifikator. Dette førte til at alle nodene fikk oppdatert sitt topologibilde innen 160μs (1 rundetid).

6.3.5.6 Bryting av en link - test 6 Ved test nummer 6, bryting av en link, var det flere usikkerheter som kom opp. På grunn av utkastets mangelfulle informasjon om wrapping, og hva som skal skje når en link brytes, måtte jeg ta noen egne beslutninger. Dette fører til at resultatet kan avvike fra det som vil være tilfelle når utkastet utvikler seg og standarden kommer

ut. Det kommer frem av teksten at det vil sendes en beskyttelsemelding fra de to nodene som er direkte påvirket av linkfeilen (de to nodene som har fått linken brutt mellom hverandre). Disse beskyttelsemeldingene vil i alle nodene de kommer til føre til sletting av de postene i topologibilde som er lengre unna enn avsendernoden. Hvis en node mottar en beskyttelsemelding som indikerer brutt link vil den slette poster i topologibildet (et topologibilde for indre og et for ytre ring) slik:

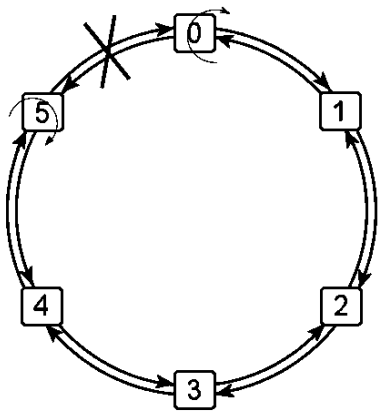
```
Indre rings topologibilde:
N = antall hopp til avsender av beskyttelsemelding
Slett alle poster som er flere enn N hopp unna

Ytre rings topologibilde:
M = antall hopp til avsender av beskyttelsemelding
Slett alle poster som er flere enn M hopp unna
```

De egne beslutninger som ble tatt var om det skulle sendes topologipakker på grunnlag av beskyttelsemeldingene som ble mottatt. Jeg valgte å teste begge mulighetene.

Resultatet ved bryting av link uten sending av topologipakker (10090μs.) gjenspeiler nodenes ventetid før timeren går ut. Slik denne testen er gjennomført sletter nodene sitt topologibilde når de mottar en beskyttelsemelding fra nodene som har merket bruddet i linken. At nodene da med en gang sender ut en topologipakke på hver ring vil korte ned oppdateringstiden. Den målte tiden ble da 300μs. Denne tiden skiller seg fra de andre testene der ringen er hel. Nodene må først slette noen poster i sitt gamle topologibilde. De vil så bygge opp, eller reparere, et nytt ut ifra den informasjonen de mottar fra de andre nodene. Det er uklart hvor mye informasjon som skal slettes i de forskjellige nodene. All naboinformasjonen som finnes i topologibildene må også oppdateres for å stemme overens med den nye topologien som nå er på ringen. For å finne ut om topologibildet er konsistent må det også defineres hva som er en konsistent tilstand.

Figur 27 på neste side viser et eksempel der det kan være usikkert hva en konsistent tilstand av topologibildet er. Utkastet spesifiserer hvordan topologibildet skal bygges opp. Hver post inneholder en node og nodens to naboer. Hvis ringen i figuren hadde vært feilfri ville naboene til node 5 være 0 og 4. Når ringen brytes vil node 5 ikke ha direkte kontakt med node 0 lenger. Hva naboene til node 5 vil være vil da kunne variere etter spesifikasjon og implementasjoner. Naboen langs den indre ringen vil enda være node 4. Naboen langs den ytre ringen kan være udefinert, noden selv (node 5), node 4 og node 0. Node 0 er ikke direkte tilknyttet men ved wrapping kan RPR-ringen likevel sees på som en logisk ring. Data kan da fortsette langs den logiske ringen etter å ha gått rundt den motsatte ring enn den de startet på (ref. figur 15).

Når nodene mottar beskyttelsemeldinger må det i tillegg til fjerning av poster i topologibilde endres noen poster. Dette gjelder de postene som tidligere hadde naboer som kunne nås over den linken som nå er brutt. I figur 27 på neste side må alle nodene endre på de postene der node 0 og 5 er. Endring av postene må gjøres for å oppdatere naboinformasjon (se forrige avsnitt). Disse postene vil bli oppdatert i de forskjellige nodene når nettverkskartlegging får utvekslet sine pakker. Simuleringer viser at disse postene også kan bli riktig oppdatert ved mottak av beskyttelsemeldinger. Tiden det vil ta før alle har oppdatert sitt topologibilde vil



Figur 27: Linken mellom node 0 og 5 er ødelagt.

være den tiden beskyttelsemeldingene bruker for å nå alle nodene på RPR-ringen. I figur 27 vil dette gjenspeile tiden det tar for node 0 å motta pakker fra node 5 og node 5 å motta pakker fra node 0 (150μs). Grunnen til at resultatet (300μs.) er betydelig høyere enn ideell tid som i denne simulatoren er 150μs. er at topologipakker må oppdatere topologibildet som ikke er konsistent etter de kravene jeg har satt. I min simulator skal nodens nabo over den brutte linken settes til "0". På grunn av at denne informasjonen må sendes til alle nodene på RPR-ringen må topologipakker sendes for at nodene skal oppdatere naboinformasjon i sine topologibilder.

6.3.5.7 Bryting av to linker - test 7 Bryting av to linker slik at det blir to separate nett vil gjøre at de ulike nodene kun kan nå et subsett av de andre nodene. Som i forrige avsnitt der én link ble brutt er det to resultat. Ett uten sending av topologipakker ved bryting av linker og ett med sending. Vi ser også her en markant forskjell mellom de to måtene å løse problemet på. Som vist i tabell 4 på side 47 er resultatet uten pakkesending relativt likt de andre endringene. Med pakkesending er tiden mye lavere. Den er også mindre enn tiden målt i forrige test. Dette vil være på grunn av at de to nettene som nå finnes er mindre enn det ene store under bryting av én link. Pakkene som må utveksles mellom nodene i nettet trenger ikke å gå så langt som de må i det ene store nettet før alle har mottatt de, og oppdatering i de to delene av nettet vil foregå parallelt.

6.3.5.8 Sammensetting av brutt nett - test 8 Etter å ha brutt to linker i RPR-ringen i forrige test så vil test 8 sette de to isolerte nettene sammen igjen. Også her er det stor forskjell på resultatet dersom det blir utløst sending av topologipakker når linken kommer opp igjen.

Oppsummering

Kapittelet har tatt for seg litt simulatorteori. Deretter ble de åtte testene som er kjørt på de ulike mekanismene presentert. Testene legger til, fjerner, endrer noder og bryter linker. Brutte linker blir også satt sammen. Darwin har like resultater på mange av testene, mens utkast 0.3/1.0

varierer mer. På grunn av usikkerheter i forslagene, blant annet på utløsing av utsending av topologipakker, er det noen av testene som har flere resultater.

7 Presentasjon av egen metode for nettverkskartlegging

Som en del av min oppgave skal jeg presentere en egen metode for nettverkskartlegging i RPR. Dette kapittelet tar for seg utviklingen av denne metoden fra designkriterier til presentasjon av hvordan den fungerer.

De forskjellige metodene for nettverkskartlegging har flere fordeler og ulemper. Min egen metode har som mål å kombinere de gode sidene ved de fremlagte mekanismene og ta inn egne punkter og forbedringer der jeg mener ting kan gjøres bedre.

7.1 Diskusjon

Ved utvikling av en metode for nettverkskartlegging er det mange forhold som må avklares. Metoden som blir gjennomgått i dette kapittelet skal derimot ikke bygges opp fra bunnen slik at noen av delene vil tas rett fra de fremlagte forslagene. Ved å ta hensyn til styrkene og svakhetene av de fremlagte forslagene kan man kombinere de to for å finne en bra alternativ løsning på nettverkskartlegging.

7.2 Designkriterier

Basert på forslagenes styrker og svakheter og erfaringer ved nettverkskartlegging i RPR kan det settes opp noen punkter som bør gis ekstra oppmerksomhet. Dette er egenskaper som er ønskelig for en mekanisme for nettverkskartlegging. Det er derimot ikke sikkert at metoden støtter alle de nevnte punktene når den står ferdig. Kriteriene har som hensikt å finne en løsning på nettverkskartlegging. Jo flere av kriteriene som blir oppfylt desto bedre er mekanismen dokumentert. Dette betyr derimot ikke at mekanismen er fullstendig hvis alle punktene er oppfylt.

De nummererte punktene er ikke prioritert på noen måte. Nummerene er kun for å kunne referere til de senere. Noen av punktene inneholder også en forklaring for å tydeliggjøre målet i kriteriet.

1. **Mål:** Alle noder skal før eller siden, uansett tap av en eller flere topologipakker, få et korrekt topologibilde av RPR-ringen.

2. **Mål:** Det må defineres om nodene skal gi bort informasjon om seg selv til andre eller hente inn informasjon fra de andre nodene.

   Forklaring: Darwin-forslaget henter inn informasjon fra de andre nodene. Topologipakken Darwin sender ut samler informasjon om alle de andre nodene på RPR-ringen. De andre nodene drar ikke nytte av informasjonen i avsendernodens pakker slik som mekanismen i utkast 0.3/1.0 gjør. Mekanismen i utkast 0.3/1.0 bruker informasjonen andre sender ut for å bygge opp topologibildet.

3. **Mål:** Topologibildet bør kunne sjekkes for konsistens.

   Forklaring: Dette gjøres enten ved at det bygges opp med redundant informasjon for feilsjekk, eller ved at det finnes en annen måte å sjekke topologibildet. Denne delen av mekanismen er grundigere forklart i punkt 5.3.1 på side 27.

4. **Mål:** Innholdet i topologipakkene må defineres.

   Forklaring: Informasjonen topologipakkene skal inneholde må bestemmes ut fra hva mekanismen skal gjøre. Innholdet i topologipakkene kan enten brukes av mekanismen for nettverkskartlegging eller andre mekanismer på RPR-ringen (for eksempel fairness). Hvis topologipakkene skal inneholde felt som ikke skal brukes av nettverkskartlegging må dette være i samsvar med hva arbeidsgruppen har definert.

5. **Mål:** Topologipakker som sirkulerer på ringen bør ikke ha noe å si for mekanismens funksjon.

   Forklaring: Hvis en node feiler etter å ha sendt ut sine topologipakker vil pakkene sirkulere på ringen til TTL-feltet i pakkene blir 0. Disse pakkene må ikke føre til at det blir feil i noen av nodenes topologibilde.

   Dette problemet reduseres ved å kun la pakkene samle informasjon om de andre nodene slik som i Darwin eller justere TTL-verdier slik at dette problemet ikke oppstår.

6. **Mål:** Mekanismen bør velge fornuftige verdier på TTL i topologipakkene den sender ut. Dette kommer av punkt 5.

7. **Mål:** Mekanismen skal kunne støtte regelmessig utløsing av pakkesending og være koordinert.

   Forklaring: Dette er punkter som begge handler om når topologipakker skal sendes som igjen fører til at topologibildet blir oppdatert ved en endring. Regelmessig og koordinert utløsing av pakkesending gjør tilsammen at alle nodene på RPR-ringen så fort som mulig oppdager en endring i nettets topologi.

8. **Mål:** Mekanismen og oppbygning av topologibilder må støtte metoder for å fjerne, legge til og endre informasjon lagret i det lokale topologibildet slik at det hele tiden gjenspeiler den virkelige topologien som finnes på RPR-ringen.

7.3 Beskrivelse av mekanismen for nettverkskartlegging

Mekanismen har et hovedprinsipp som er likt prinsippet i Darwin. Informasjon samles inn fra de andre nodene for å bygge opp sitt eget topologibilde. Nodene bruker derimot informasjon de andre sender ut i en konsistenssjekk for å sjekke om det topologibildet de har er riktig.

Mekanismen går slik:
Hver node sender regelmessige topologipakker ut på begge ringene. Disse pakkene inneholder nodens MAC-adresse og nodens MAC-tilleggsinformasjon. Som i Darwin er MAC-adressen og MAC-tilleggsinformasjoen tilsammen kalt MAC-bindingene (ref. kapittel 5.2.1 på side 26). Hver node legger til sine bindinger på slutten av andres topologipakker når de ankommer noden. Noden vil også bruke pakkene til å verifisere sitt eget topologibilde etterhvert som den mottar topologipakker fra de andre nodene.

7.3.1 Variabelen STATUS

En variabel (STATUS) settes når noden selv tror den har et komplett topologibilde. Når denne er satt sjekker noden alle innkommende topologipakker om antall hopp til avsender stemmer med den informasjonen som finnes i det lokale topologibildet. Det vil ikke være nødvendig å sjekke topologibildet for konsistens da noden vet at topologibildet den har ikke kan være rett. Noden kan aldri være sikker på å ha rett topologibilde, men den kan være sikker på at den ikke har rett topologibilde. Derfor setter den statusvariabelen når den tror den har rett topologibilde. Videre bruk av variabelen blir tatt opp i de følgene kapitlene.

7.3.2 Oppstart

Ved oppstart setter noden variabelen STATUS til NOTOK. Noden sender så en topologipakke på hver ring med TTL lik maksimal ringstørrelse (255) og nullstiller topologitimeren.

7.3.3 Mottak av topologipakke på samme ring som den ble sendt

Når en node mottar en topologipakke den ikke selv har sendt legger den til sine MAC-bindinger på slutten av pakken. Hvis STATUS er satt til OK i denne noden, regn ut antall hopp til avsendernoden på grunnlag av TTL i den mottatte topologipakken. Hvordan dette gjøres er tidligere forklart i kapittel 5.3.6.1 på side 30. Den mottatte pakken videresendes før det lokale topologibildet sjekkes for å se om noden finnes og at antall hopp lagret i topologibildet er lik den utregnede verdien. Hvis verdiene ikke er like vil dette bety at nodens topologibilde ikke er rett eller at det har skjedd en feil under sending av topologipakkene. Hvis de ikke er like, sett derfor STATUS til NOTOK og send ut en topologipakke på hver ring for å oppdatere sitt eget topologibilde. Nullstill den periodiske timeren for sending av topologipakker.

Ved mottak av topologipakker som noden selv har sendt oppdaterer den topologibildet hvis de to pakkene begge indikerer en endring i nettets topologi. På grunn av at to pakker må komme inn til noden før en endring kan skje vil heller ikke sjekk mot topologibildet for endringer settes i gang før begge pakkene er returnert til noden. De to pakkene må også representere det samme topologibildet. Dette er tilsvarende som vist i Darwins forslag tidligere. Figur 20 på side 29 viser oppbygningen av pakkene hvis det skal kunne gjøres en endring i topologibildet.

7.3.4 Mottak av topologipakke på motsatt ring enn der den ble sendt

Hvis en node mottar en topologipakke som har opprinnelse fra den andre ringen legger nodene ikke til sine MAC-bindinger. Noden som wrapper pakken første gang på sin vei til mottakeren vil sette et bit i MAC-tilleggsinformasjon som viser at pakken ble wrappet i denne noden. Det er også et felt i topologipakkene som sier hvilken ring pakken har startet sin ferd på. Basert på denne informasjonen kan avsendernoden regne ut riktig topologibilde av ringen.

Wrapping av topologipakkene vil også føre til at konsistenssjekken ikke vil bli gjennomført etter en topologipakke er wrappet. Konsistenssjekken for pakker som blir wrappet vil bare gjøres frem til den noden som wrappet pakken første gangen. Nodene på motsatt ring ser at

pakken har opprinnelse fra den andre ringen og utelater da sjekken. Eventuelle noder som topologipakken besøker etter å ha blitt wrappet to ganger, som ligger på samme ring som den ble sendt, vil kunne se at pakken tidligere har blitt wrappet ved å gå gjennom MAC-bindingene.

7.3.5 Oppførsel ved brutt link

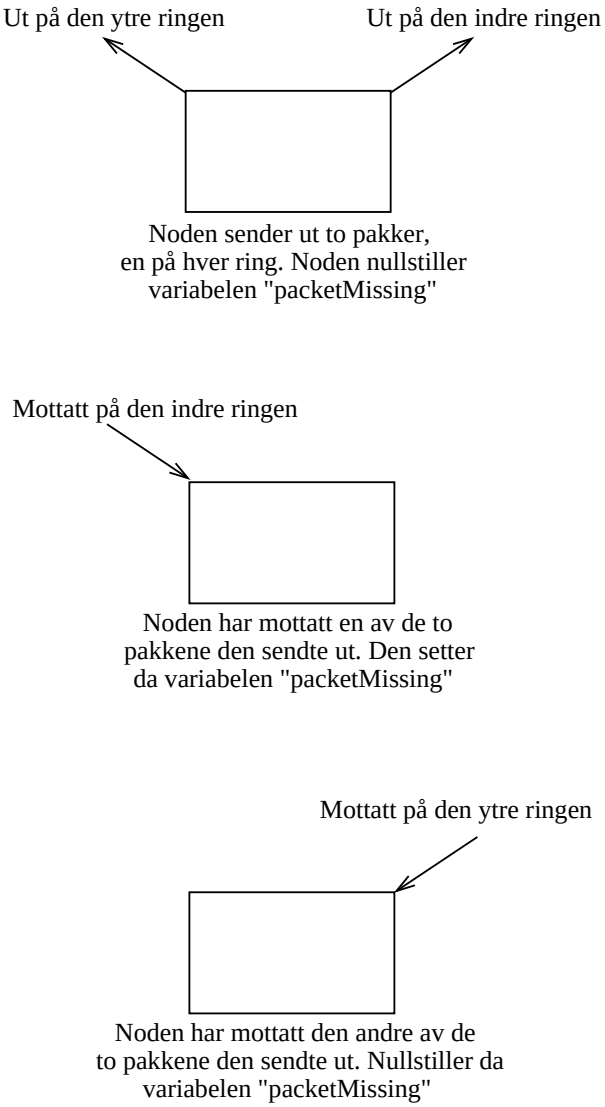
Når en link blir brutt skal nodene som er direkte tilknyttet bruddet sende ut beskyttelsemeldinger på den ringen som går bort fra feilen. Denne delen er ikke en del av mekanismen for nettverkskartlegging. Nettverkskartlegging bruker derimot disse pakkene til å rette på sitt eget topologibilde. Ved mottak av en beskyttelsemelding som indikerer brutt link vil alle nodene som har lengre avstand enn til avsendernoden bli slettet fra det lokale topologibildet. Dette vil gjøre at topologibildet gjenspeiler nettets topologi etter bruddet. Denne idéen er tatt fra forslaget i utkastet.

Ved mottak av en beskyttelsemelding vil også nodene utløse en ny runde nettverkskartlegging for å oppdatere deres topologibilde som nå mangler poster. Dette vil foregå som vanlig ved at de sender ut en topologipakke på hver ring.

7.3.6 Sending av topologipakker

Sending av topologipakker vil bli utløst av topologitimeren, konsistenssjekken og mottak av beskyttelsemelding. Når det sendes ut pakker på ringene vil timeren nullstilles og noden vil også nullstille en variabel som sier om det har kommet inn noen pakker som noden skal vente på å prosessere eller ikke. Denne variabelen brukes til å finne ut om begge topologipakkene den aktuelle noden sendte ut har returnert. Noden må ha begge disse pakkene for å kunne sjekke om topologibildet har endret seg. Hvis det skulle skje at en av disse to pakkene blir borte på veien rundt ringen vil denne variabelen indikere at det kun har returnert 1 pakke. Denne vil altså bli nullstilt når noden sender ut pakker slik at den ikke bruker en gammel pakke for å regne ut topologibildet. Figur 28 på neste side viser bruken av variabelen. “packetMissing” blir nullstilt når det sendes ut topologipakker for at systemet ikke skal komme i ubalanse i prosesseringen av to og to pakker.

Eksempel med referanser til figur 28 på neste side:
I det første bildet sender noden ut en topologipakke på hver av de to ringene. I det andre bildet har én av de to pakkene returnert til avsendernoden. Variabelen settes da for å indikere at kun 1 har kommet tilbake. Hvis den andre topologipakken blir tapt vil variabelen indikere at det mangler en pakke. Derfor vil ikke topologibildet bli oppdatert. Når noden skal sende ut topologipakker neste gang vil variabelen fremdeles indikere at det mangler en pakke. Hvis de to nye pakkene hadde blitt sendt ut på ringen uten å gjøre noe med variabelen ville den ene gamle og en av de to nye prosesseres. Vi vil unngå dette og resetter derfor variabelen slik at noden ikke mangler noen pakker når den sender ut nye. Den første av de to nye som kommer tilbake til avsender vil da sette variabelen igjen. Den andre pakken vil føre til at topologibildet beregnes på grunnlag av de to nye topologipakkene.



Figur 28: Figuren illustrerer bruken av en variabel som holder en del av tilstanden til nodens nettverkskartlegging. “packetMissing” brukes for å indikere om noden mangler noen topologipakker eller ikke.

7.3.7 Topologi timer

Mekanismen vil prestere bedre jo mere spredt de forskjellige nodene sender ut sine topologipakker på ringene. I noen tilfeller er det den første noden som oppdager endring i nettets topologi som da "sier fra" til de andre nodene at det har skjedd en endring. I tilfeller der en node "sier fra" til de andre vil alle nodene sende sine pakker nesten samtidig. Hvis man fra dette punktet hadde en fast tid for timeren ville det fra denne tiden bli lite spredning i utsending av topologipakker. Derfor vil topologi timeren være tilfeldig rundt en fastsatt middelverdi. Denne middelverdien må settes til en verdi der mekanismen kan klare å opprettholde kravet om 50ms før en ny rute er satt opp etter en feil på RPR-ringen. For å ha en verdi å starte testene med vil den i simuleringene settes til 20ms med et minimum på 75% og et maksimum på 125% av denne. En ny verdi genereres etter hver runde nettverkskartlegging i alle nodene og verdiene i de forskjellige nodene har ingenting med hverandre å gjøre.

Oppsummering

Kapittelet startet med å presentere åtte designkriterier. Disse kriteriene kommer av erfaringer med nettverkskartlegging i RPR. Deretter ble mekanismen presentert med variabelen STATUS og konsistenssjekken som viktige punkt. Mekanismen bruker samme prinsipp som Darwin-forslaget.

8 Simulering og drøfting av egen mekanisme

Dette kapittelet vil ta for seg simuleringen av mekanismen for nettverkskartlegging som ble presentert i forrige kapittel. Første del presenterer hvordan mekanismen forventes å fungere. I andre del vil problemer som har kommet opp under simuleringene bli presentert. Deretter vil resultatene fra testene presenteres og drøftes. Til slutt vil mekanismens virkemåte sammenlignes med designkriteriene for å se om disse er oppfylt.

8.1 Hvordan mekanismen er antatt å fungere

Denne delen vil ta for seg hvordan mekanismen er antatt å fungere under ulike hendelser som skjer på RPR-ringen.

8.1.1 Ved oppstart

Oppstart kan være om det er oppstart av ringen eller oppstart av mekanismen i en node når denne blir lagt til på ringen. I disse tilfellene vil nodene som starter opp sende ut topologipakker på begge ringene for å oppdatere sitt eget topologibilde. Dette vil også føre til at de andre nodene mottar topologipakker fra en node de ikke har i sitt topologibilde. På grunn av dette vil det utløses pakkesending i alle nodene på RPR-ringen, noe som igjen fører til at alle får oppdatert sitt topologibilde innen tiden det tar for pakkene å gå rundt ringen. I den første testen sender alle ut sine pakker samtidig. Derfor vil alle nodene få oppdatert topologibildet på tilnærmet samme tid. Ved tillegging av en node vil noden “si ifra” til de andre nodene. Noden på motsatt side av ringen er den noden som senest finner ut at det har blitt en endring fordi pakkene bruker lengst tid frem til denne. Tiden for oppdatering av topologibildet vil da bli 1,5 rundetid.

8.1.2 Fjerning av en node, ingen wrapping

Ved tillegging vil den nye noden selv sende ut pakker som utløser sending av pakker også hos de andre nodene. Det vil derimot ikke bli utløst noen sending av pakker hvis en node blir fjernet. Hvis mekanismen uavhengig av andre mekanismer skal finne ut at en node har blitt fjernet er resultatene av denne testen mere varierende enn ved tillegging. Nodene har ulike tidspunkter for periodisk nettverkskartlegging som forklart i punkt 7.3.6 på side 56. Dette vil være en fordel i denne situasjonen fordi det vil være gjennomsnittlig kortere tid til neste node sender ut topologipakker enn gjennomsnittstiden hvis alle sendte samtidig. Hvilken node som først sender ut sine pakker etter at en node har blitt fjernet vil være tilfeldig siden fjerningen av noden kommer på et vilkårlig tidspunkt. Pakkene denne sender ut vil på grunn av konsistenssjekken utløse pakkesending i de andre nodene. Dette vil føre til at alle nodene på ringen vil oppdatere sitt topologibilde. Hvis det er flere noder på ringen vil den gjennomsnittlige tiden til neste node sender ut sine pakker bli mindre. Dette vil kunne føre til at jo flere noder det er på ringen jo kortere tid vil det ta for alle nodene å få oppdatert sin topologi. Dette vil også kunne avhenge av den fysiske størrelsen på ringen.

8.1.3 Endring av identifikator på en node

Ved endring av identifikator på en node vil det være en identifikator i alle topologibildene rundt ringen som er annerledes etter noden har skiftet. Nodens nye identifikator vil være den som utløser en oppdatering i topologibildet i de andre nodene. Nodene vil da kunne bli oppdatert innenfor samme tidsintervall som under fjerning av en node.

8.1.4 Bryting av link, wrapping

Jeg skal også teste mekanismen ved å wrappe RPR-ringen på en og to steder. Designet av mekanismen sier at nettverkskartlegging skal utløse utsending av topologipakker basert på beskyttelsemeldinger som mottas. Konsistenssjekken vil ikke gjennomføres etter topologipakkene har blitt wrappet første gangen og vil derfor ikke påvirke oppdateringstiden.

8.1.5 Sammensetting av brutt, wrappet link

Når de to nettene som ble delt i forrige punkt blir satt sammen igjen vil konsistenssjekken fungere bedre. Topologibildet som nodene har å sjekke mot vil være annerledes enn det topologibildet de neste pakkene som kommer inn indikerer. Dette vil antakelig fungere slik at det gjør at sammensetting av en brutt link er en hendelse som nettverkskartlegging raskt vil kunne håndtere. Tiden for oppdatering i alle nodene vil kunne få resultat som ligger rundt resulatene for tillegging av en node på ringen. Det er som i det tidligere punktet at det er tilfeldig når neste node utløser pakkesending etter RPR-ringen er rettet opp, men denne vil igjen utløse pakkesending i alle de andre nodene. Det vil også være mulig å utløse utsending av topologipakker basert på eventuelle beskyttelsemeldinger som blir sendt rundt RPR-ringen etter sammensetting. Dette vil kunne gi en oppdateringstid som er kortere enn uten beskyttelsemeldingene.

8.2 Testkjøringer under implementering

I denne delen er det tatt opp temaer som har kommet opp under implementeringen av min egen metode. Dette er problemer som jeg har tatt hensyn til under programmeringen frem til den ferdige simulatormodellen. Det opprinnelige designet av simulatoren har blitt endret etterhvert som testsimuleringer kjøres. Dette for å forsøke å ende opp med en mekanisme for nettverkskartlegging som presterer bra på alle felt.

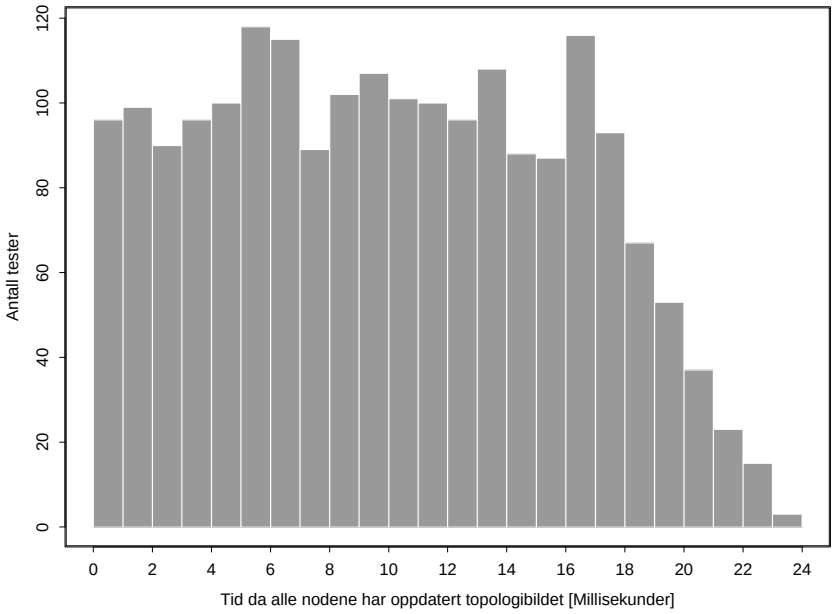
8.2.1 Tapt topologipakke

Hvis en eller begge av pakkene en node sender ut blir tapt kan dette føre til lengre periode med feil topologibilde. Dette vil komme av at node kan ikke oppdatere sitt topologibilde hvis kun én av to pakker returnerer til avsenderen.

8.2.2 Endring av identifikator på en node

Ved endring av identifikator på en node vil ikke avstanden til de andre nodene på RPR-ringen endre seg. Dette fører til at det ikke blir utløst pakkesending i nodene før noden som har endret

identifikator sender ut sine pakker. Slik det var i det opprinnelige designet sendte ikke noden som endret sin identifikator ut topologipakker på ringene før timeren gikk ut. De andre nodene ble da nødt til å vente til sin timer gikk ut eller "den nye" nodens timer gikk ut før de kunne få oppdatert sitt topologibilde. Tiden før alle nodene har fått oppdatert sitt topologibilde vil da kunne ta opp til en hel timerperiode for nettverkskartlegging. Figur 29 viser fordelingen av resultatet med det opprinnelige designet uten utløsing av pakkesending i noden som endret identifikator. Det er kjørt 2000 tester med tilfeldige verdier på periodisk timerverdi og tiden da noden endret identifikator. Vi kan se at de resultatene er relativt jevnt fordelt opp til circa 20ms som er gjennomsnittstiden for timerperioden. På grunn av at timeren varierer rundt denne gjennomsnittstiden er det også noen tilfeller der det har tatt lengre tid enn 20ms.



Figur 29: Bildet viser fordelingen av resultat ved endring av identifikator på en node. Diagrammet er bygd opp som et histogram med 1 millisekund intervall.

For å gjøre denne delen av mekanismen bedre skal nodene sende ut topologipakker hvis den endrer identifikator. Når de andre nodene da oppdager at det har kommet en ny node inn på RPR-ringen vil de sette i gang en runde nettverkskartlegging selv.

8.2.3 Konsistenssjekk

Forbipasserende topologipakker skal sjekkes mot topologibildet for konsistens. Slik det er nå vil nodene ta vare på den korteste avstanden i sitt topologibilde. Det vil derimot komme to pakker med to forskjellige avstander til hver enkelt node, en for hver ring. Kun i særtilfeller vil de to pakkene indikere samme antall hopp til avsendernoden (hvis det er like antall noder på ringen og avsenderen er noden på motsatt side av ringen). Dette må tas hensyn til slik at ikke nodene tror det er feil på ringen under prosessering av halvparten av topologipakkene.

Siden nodene tror topologibildet er komplett vil de ha en formening om hvor mange noder det finnes på ringen. Dette vil være en del av topologibildet. Løsningen på problemet over vil da være at nodene regner ut hvor lang den lange veien skal være med antall noder på ringen og den korteste veien som er lagret i topologibildet fra før.

8.2.4 Fjerning av en node, ingen wrapping

Fjerning av en node kan i prinsippet gjøres på flere måter. Først fjernet jeg en node ved å koble sammen de to nodene på hver side av den som skulle fjernes. I min simulator var det node nummer 0 som skulle fjernes. Med 16 noder på ringen ble node 15 og node 1 koblet sammen. Disse var de nærmeste naboene til node 0. Denne testen sier noe om hvordan mekanismen skal oppdage endring av nettets topologi uten at noen av nodene får beskjed fra noen andre mekanismer. Det vil altså ikke genereres noen beskyttelsemeldinger som kan utløse nettverkskartlegging. Dette krever at nodenes regelmessige utsending av topologipakker er nødt til å oppdage nettets endring. Det vil være tilfeldig når de neste topologipakkene sendes ut på ringene av en av nodene som er igjen på RPR-ringen. Disse pakkene vil føre til at nodene oppdager en inkonsistens i deres topologibilde. De vil derfor oppdatere sitt eget topologibilde. Hvis antallet noder på ringen økes minskes den gjennomsnittlige tiden til hvor lenge det er til den første noden sender ut pakker.

En annen situasjon som kan oppstå ved fjerning er at en node går ned eller blir fjernet fra ringen da den prosesserer andre noders topologipakker slik at disse blir tapt. Avhengig om noden som sendte ut pakkene har et topologibilde fra før eller ikke kan dette forsinke i ulik grad dens oppdatering av sitt topologibilde.

8.2.5 Bryting av linker, wrapping

Hypotesen på dette punktet viste seg å ikke være helt riktig. I det opprinnelige designet var det slik at nodene ikke sjekket topologibildets konsistens når pakkene hadde blitt wrappet 1 gang (topologipakkene befinner seg på motsatt ring enn den de ble sendt ut på). Dette fører til at nodene vil kun sjekke konsistens med den pakken som ikke har kommet forbi det punktet der en link har blitt brutt. Konsistenssjekken har i dette punktet ingen funksjon da antall hopp langs veien utenom det wrappede punktet ikke har endret seg. En mulig løsning på dette vil være å ikke telle ned TTL når topologipakkene befinner seg på motsatt ring enn den de startet på. Som sagt tidligere foreslår senere utkast å ikke telle ned TTL på motsatt ring. Ved å ikke telle ned TTL på motsatt ring kan man skjule for konsistenssjekken at pakkene har gått en runde rundt den andre ringen. Dette vil fungere hvis det ved wrapping har blitt fjernet en node. I testen ble en link mellom to noder brutt. Konsistenssjekken vil ikke bli utløst uansett når en link blir brutt og ingen node blir fjernet. Dette kommer av TTL-verdien som da ikke endrer seg når noden går forbi det brutte punktet. Selv om en link blir brutt er det åpenbart interessant for nodene å få vite om feilen. Dette vil skje når de mottar sine egne topologiopakker når de har gått rundt RPR-ringen.

8.2.6 Sirkulerende topologipakker på ringen

Hvis en node blir fjernet fra ringen i tidpunktet mellom den sender ut topologipakker og den mottar de samme pakkene vil det kunne oppstå problemer. Pakkene må sirkulere på ringen til TTL-verdien blir 0. Hvis da denne noden er fjernet fra de andre nodenes topologibilder, men disse nodenes topologibilder er riktige vil det føre til at STATUS blir satt til NOTOK. Dette vil kunne skje flere ganger selv om nodene allerede har et topologibilde som stemmer med virkeligheten. Den eneste ulempen med dette er at det vil utløses utsending av topologipakker i alle nodene de aktuelle pakkene kommer til. Dette er topologipakker som ikke trengs og kun vil føre til mer last på RPR-ringen. Hvis en node fra før har korrekt topologibilde vil de nye pakkene den sender ut ikke gjøre noen endringer i bildet den har.

8.3 Resultat

For å kunne sammenligne min egen metodes ytelse med de fremlagte forslagene har jeg kjørt de samme testene på denne. Resultatene er først presentert i to tabeller for å drøftes senere. Det er en tabell med resultatene fra kjøringer med 16 noder og en fra kjøringer med 255 noder. Testene ble beskrevet i tabell 2 på side 35.

I likhet med simulatormodellene for utkastet og Darwin er også denne modellen på circa 4000 linjer kode. Det vil derfor også her være mulighet for feil i kildekoden som fører til et resultat som skiller seg fra det som ville skjedd på en virkelig RPR-ring. Simulatormodellen i sin helhet, og de ulike klassene som modellen er bygget opp av er testet mye i tillegg til at resultatene som kommer ut er fornuftige. Dette indikerer derimot at kildekoden er riktig.

Test nr.	Resultat 16 noder	Kort kommentar
1	1 RTT / 160μs.	
2	circa 1 topologi timer + 1 RTT.	
3	circa 1,5 ms ($\approx$9,5 RTT)	Avhenger av størrelse på ringen, antall noder som er koblet til ringen og topologitimeren. Resultatet er tilfeldig der gjennomsnittstiden varierer med de nevnte parametere. Dette resultatet er gjennomsnitt av 2000 beregninger med simulatoren
4	1,5 RTT / 240μs.	
5	1,5 RTT	Etter endringen vil denne oppføre seg på samme måte som å legge til en node.
6	340μs. ($\approx$2 RTT)	Fast resultat på alle målinger
7	230μs. ($\approx$1,5 RTT)	Fast resultat på alle målinger
8	4337μs. ($\approx$27 RTT)	Dette er gjennomsnittstid. Figur 34 på side 69 viser den fullstendige fordelingen.

Tabell 5: Tabellen viser resultatet fra testene kjørt med mitt eget forslag med 16 noder på RPR-ringen.

8.4 Drøfting

8.4.1 Oppstart - test 1

Test 1 er som sagt ingen reell situasjon. Resultatet i denne testen er som i de andre mekanismene. Dette kommer av at her, som i de to andre testede mekanismene, starter alle nodene på RPR-ringen en oppdatering av sitt eget topologibilde samtidig ved tiden 0. Dette fører til at pakkene trenger en rundetid for å komme tilbake til avsenderen der de kan prosesseres.

8.4.2 Fjernet node ved oppstart - test 2

Test 2 vil gjenspeile noe av hendelsesforløpet når topologipakker tapes. Grunnen til at det tar så lang tid å få gjennomført nettverkskartlegging er nettopp at topologipakkene har gått tapt. I et spesielt tilfelle vil det være mulighet for at én node mister begge sine pakker. Det vil da ikke være mulig for denne noden å bygge opp et topologibilde denne runden. Noden må da vente til neste gang den sender ut pakker for å oppdatere sitt topologibilde. I testen har nodene også ikke noe topologibilde i utgangspunktet, men man kan også tenke seg situasjonen

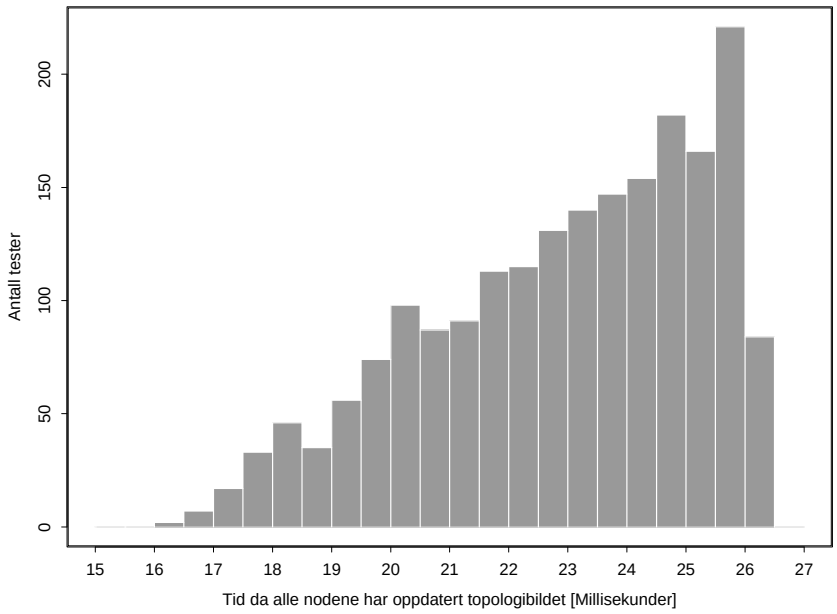
Test nr.	Resultat 255 noder	Kort kommentar
1	1 RTT / 2550μs.	
2	22,9ms (≈9 RTT)	Gjennomsnittsverdi der verdiene ligger mellom 16,2ms og 26,5ms
3	5375μs. (≈2 RTT)	Dette er gjennomsnittsverdi. Resultatene er presentert i diagram senere.
4	1,5 RTT / 3825μs.	
5	1,5 RTT	Etter endringen vil denne oppføre seg på samme måte som å legge til en node.
6	5300μs. (≈2 RTT)	Fast resultat på alle målinger
7	8275μs. (≈3 RTT)	Dette er gjennomsnittstiden målt i den fullstendige fordelingen i figur 33 på side 69. Resultatet skiller seg fra den samme testen med 16 noder. Dette er nøyere forklart under drøftingen.
8	17500μs / 15900μs. (≈7 RTT / ≈6 RTT)	Dette er gjennomsnittstid henholdsvis uten og med sending av beskyttelsemeldinger som utløser oppdatering i nodene

Tabell 6: Tabellen viser resultatet fra testene kjørt med mitt eget forslag med 255 noder på RPR-ringen.

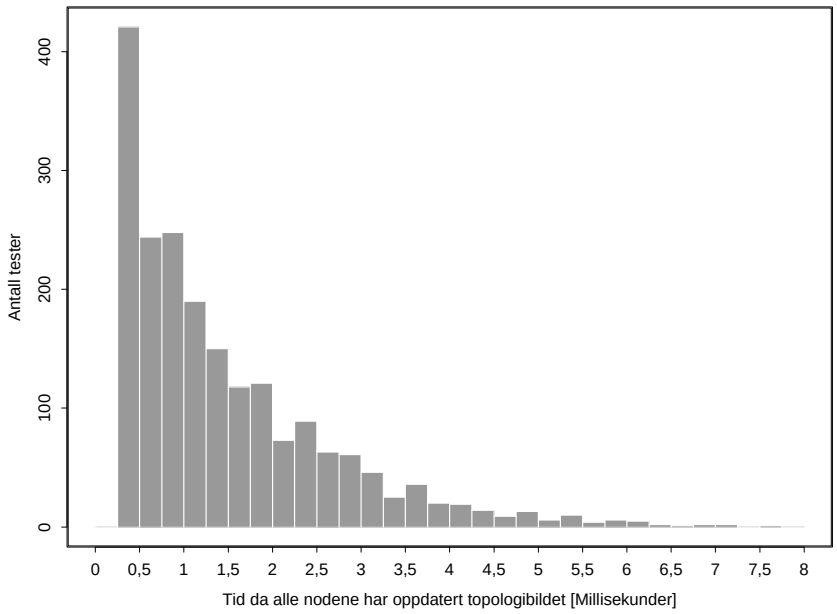
der nodene har feil topologibilde av ringen på grunn av en endring i nettets topologi. Hvis en node mister sine pakker må den vente til timeren har løpt ut før den kan sjekke nettets topologi igjen. Dette gjelder også hvis en node mister en av to pakker på vei rundt ringen. Mekanismen krever at de to pakkene kommer tilbake før noden kan bygge opp eller endre på et topologibilde. Resultatene fra test 2 er litt annerledes med 255 noder enn de er med 16. Nodenes topologitimer vil starte samtidig første runden og ligge mellom 15 og 25ms. På grunn av RPR-ringens rundetid ligger resultatene litt over disse tidene. Den komplette fordelingen av resultatene er vist i figur 30.

8.4.3 Fjerning av node - test 3

Fjerning av en node med 16 noder på RPR-ringen, som er testet i test 3, følger hypotesene. Vi ser av grafen i figur 31 på neste side at de fleste resultatene under de 2000 målingene har et resultat som gir oppdatering av alle topologibilder på kort tid i forhold til gjennomsnittstiden

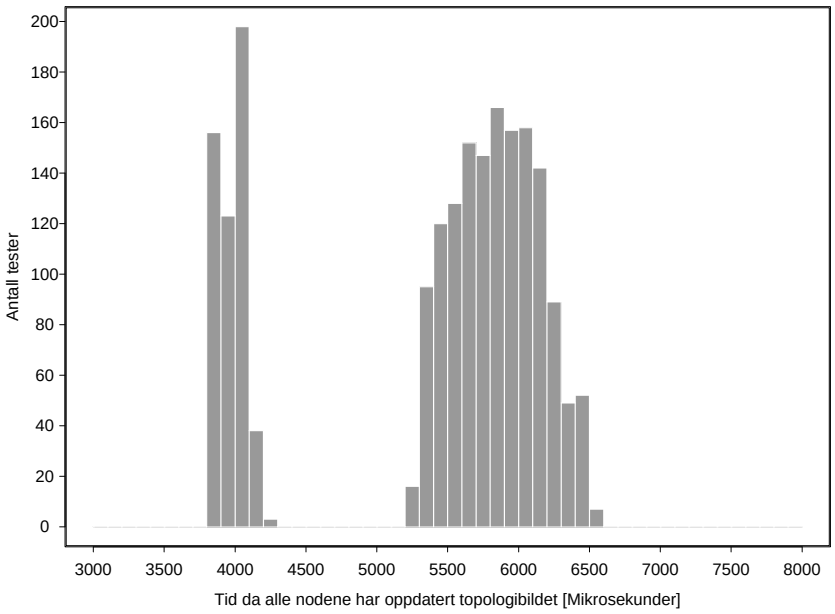


Figur 30: Bildet viser fordelingen av resultatene under test 2 (fjerning under oppstart) med 255 noder på RPR-ringen. Den største verdien målt er ≈26,5ms, mens den minste er ≈16,2ms.



Figur 31: Bildet viser fordelingen av resultatene ved fjerning av en node i en RPR-ring med 16 noder. Den største verdien målt er 7560μs, mens den minste er 250μs.

for topologitimeren som er 20ms. Den høyeste verdien målt for oppdatering er circa 7ms. På grunn av at grafen er sterk synkende vil medianen av de målte verdiene være 1125μs. Fjerning av en node da det er 255 noder på ringen vil prestere ganske annerledes. Vi ser av figur 32 fordelingen av resultatene.



Figur 32: Bildet viser fordelingen av resultat ved fjerning av en node i en RPR-ring med 255 noder. De to toppene er circa en rundetid fra hverandre. Den største verdien målt er 6580μs, mens den minste verdien målt er 3830μs.

Det som har skjedd her er at resultatene har kommet i to puljer. Ettersom nodenes timerperiode går ut vil de automatisk sende ut topologipakker. Disse pakkene vil oppdage en endring i nettets topologi både i noden de er sendt fra og i noder langs RPR-ringen. Hvilken node som først sender ut topologipakker etter fjerningen, og hvor lang tid det tar før dette skjer er tilfeldig. Fra en node oppdager en endring til den har oppdatert sitt topologibilde vil det ta 1 rundetid. Vi kan se at tidsforskjellen mellom sentrum av de to puljene er litt mindre enn 1 rundetid (2550μs). Dette betyr at i de tilfellene der resultatene ikke faller i den første puljen vil noen av nodene ha oppdaget endringen rett før den første puljen. Den største verdien i hver av puljene er veldig nær 1 rundetid fra hverandre. Dette indikerer at de siste av nodene som har fått nytt topologibilde i den første puljen utløser sending av topologipakker i en eller flere noder. Disse må så vente en rundetid for at topologipakkene de sender ut skal returnere.

8.4.4 Lagt til node - test 4, og endring av identifikator - test 5

Ved tillegging av en node og endring av identifikator vil noden selv sende ut pakker på begge ringer som vil utløse en fullstendig oppdatering av topologibildet hos alle de andre nodene på ringen. Denne delen av mekanismen er veldig stabil på 1,5 rundetid. Dette gjelder for testene

med både 16 og 255 noder på RPR-ringen. Noden lengst unna den nye er en halv rundetid unna, og det er disse som sist oppdager en endring i nettets topologi. Det vil da ta en hel rundetid til den tar imot sine egne pakker etter utsending og får oppdatert topologibildet.

8.4.5 Bryting av en link - test 6, og bryting av to linker - test 7

Tiden for oppdatering etter bryting av både en og to linker var vanskeligere å forutse enn noen av de andre resultatene. Dette kom av at det var vanskelig å fastslå når nodene fikk sendt ut sine topologipakker og når de forskjellige nodene satt seg selv i hvilken status (OK / NOTOK). Hvilken status nodene er i til enhver tid vil også bestemme om de vil gjennomføre konsistenssjekk. Slik mekanismen fungerer med utløsing av pakkesending etter bryting av en og to linker vil mekanismen gi faste tider på henholdsvis 340 og 230 μs. med 16 noder på RPR-ringen.

Med 255 noder på RPR-ringen artet oppdateringen i de forskjellige nodene seg ganske forskjellig ved bryting av en og to linker. Bryting av 1 link førte til et stabilt resultat på oppdatering i alle noder på circa 5,3ms. Ved å sammenligne dette resultatet fra resultatet målt med 16 noder på ringen kan vi se at det er et forhold mellom verdiene som er tilnærmet det samme:

$$\frac{255 noder}{16 noder} \approx \frac{5300 \mu s}{340 \mu s} \approx 16$$

Dette avhenger også av rundetiden. I dette tilfellet var linkene mellom nodene like lange i testene med 16 og 255 noder noe som ga et tilsvarende proporsjonalt forhold som over.

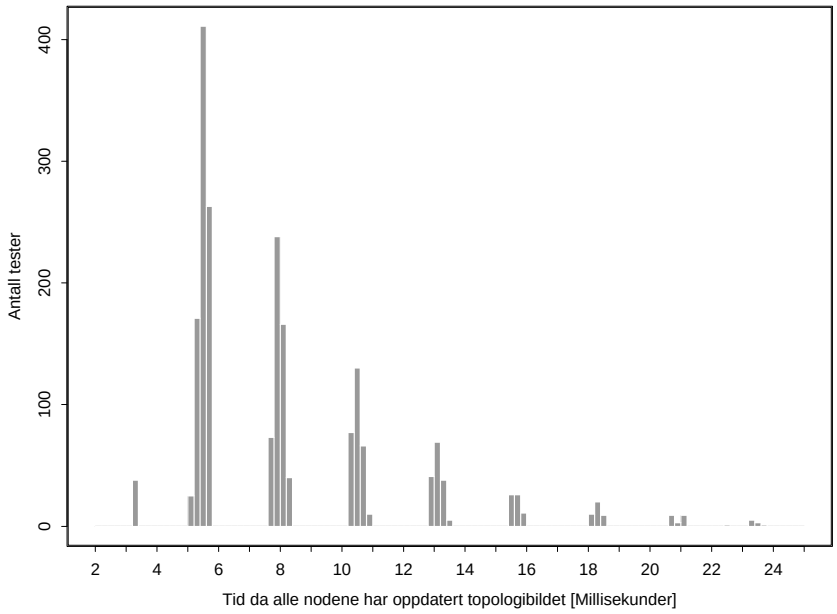
Resultatene ved bryting av to linker i en RPR-ring med 255 noder er presentert i figur 33 på neste side. Vi ser her at tidene da alle nodene har oppdatert sitt topologibilde kommer i intervaller med flest målinger på lave verdier. Intervallene er circa 1 rundetid fra hverandre. En rundetid når to linker er brutt vil i dette tilfellet være tilnærmet lik rundetiden før noen av linkene ble brutt (2550μs). Dette kommer av at nodene sier ifra til hverandre at de må oppdatere sitt topologibilde. Dette skjer kun hvis topologipakkene er på den ringen de ble sendt og ikke tidligere har blitt wrappet.

Oppdateringstiden med 16 noder var circa 1,5 rundetid. Med 255 noder på ringen var resultatet på circa 3 rundetider. Grunnen til dette vil bli tatt opp under drøftingen av sammensetting av de to nettene da det samme problemet oppstår da.

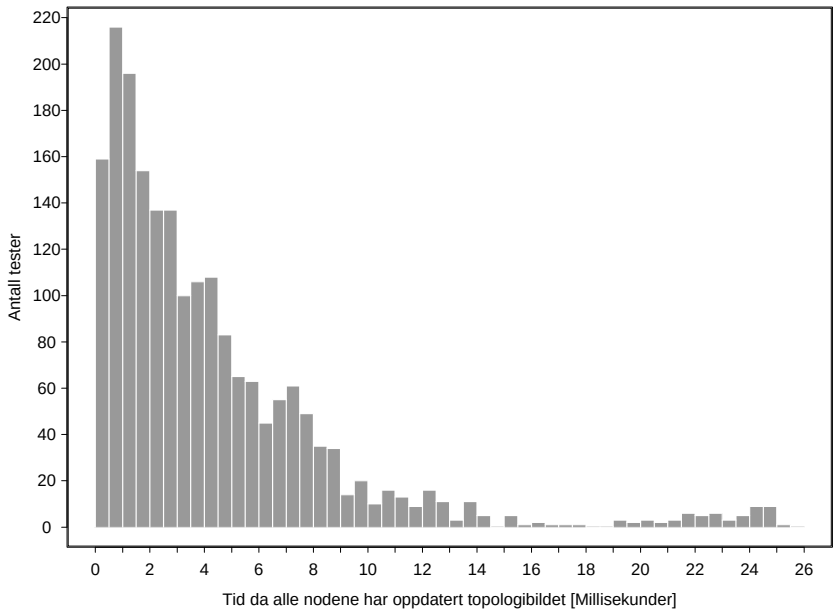
8.4.6 Sammensetting av brutt nett - test 8

Sammensetting av de to nettene er for 16 noder testet uten utløsing av pakkesending. Denne delen ligner litt på fjerning av en node uten utløst pakkesending. Figur 34 på neste side viser det fullstendige resultatet av de 2000 kjørte testene i en RPR-ring med 16 noder.

De fleste målingene oppnår en relativt raskt oppdatering. Vi ser at det er en liten andel av testene som har tider opp mot største mulige topologitimer (25000μs.). Det som har skjedd her er at en eller begge av topologipakkene til en eller flere noder er på den motsatte ringen fra den de startet på akkurat da nettene blir slått i sammen. Dette vil føre til at denne/disse pakkene ikke vil kunne bli brukt hos avsendernoden til å kalkulere et topologibilde. De vil bli nødt til å sirkulere til TTL blir null på den andre ringen. Mekanismen krever at to pakker indikerer samme endrede topologibilde og blir da nødt til å vente en hel timerperiode før den sender



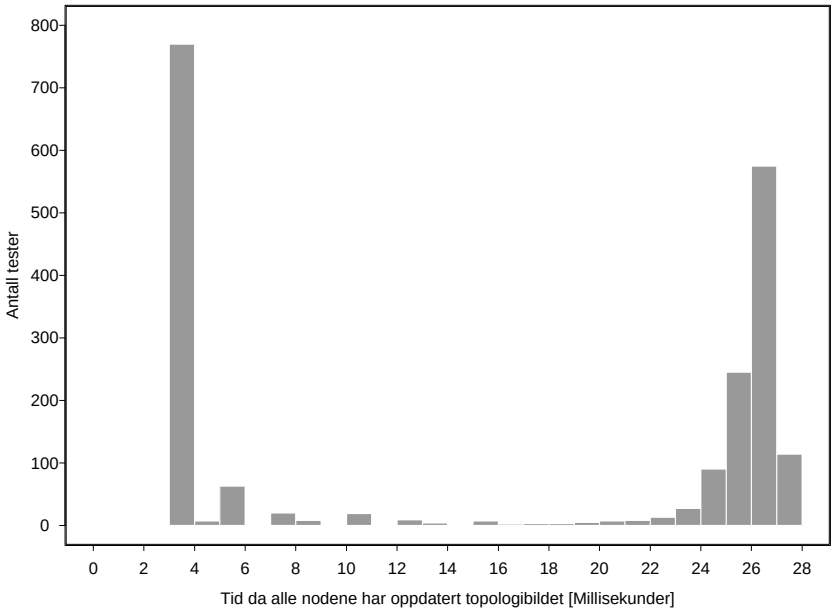
Figur 33: Her ser vi resultatet fra testene kjørt med bryting av to linker i en RPR-ring med 255 noder. Det er en rundetid mellom de ulike puljene. Den største verdien målt er ≈23,7ms, mens den minste er ≈3,2ms.



Figur 34: Bildet viser fordelingen av resultatene ved sammensetting av de to isolerte nettene. Det er 16 noder på RPR-ringen. Den største verdien målt er ≈25,1ms, mens den minste er 230μs.

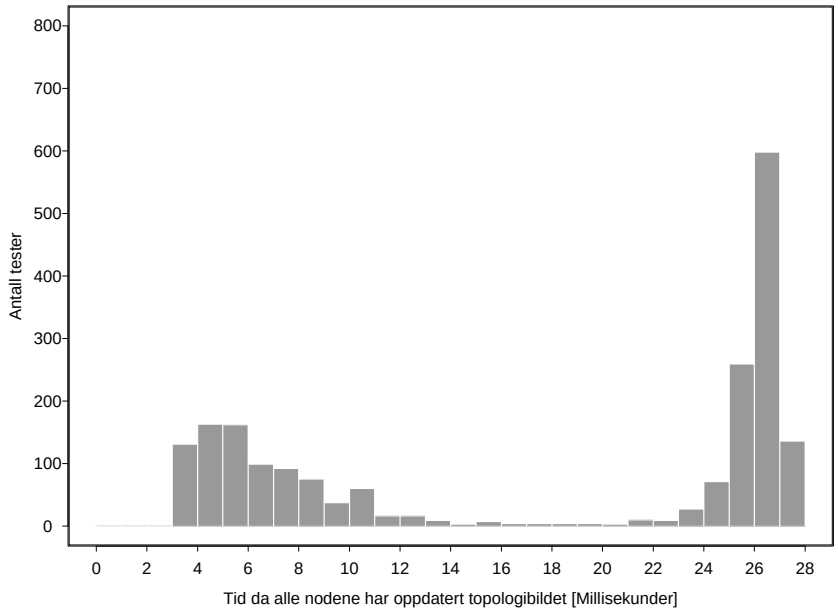
pakker på nytt. Avhengig av om det blir sendt ut beskyttelsemeldinger av beskyttelsemekanismen eller ikke ved denne hendelsen vil mekanismen for nettverkskartlegging kunne utløse utsending av topologipakker på grunnlag av disse. Dette vil føre til at oppdateringstidene blir mer deterministiske da man vet nøyaktig når nodene sender ut topologipakker. Grunnen til at det er gjort slik er for å vise at mekanismen for nettverkskartlegging ikke trenger hjelp av noen annen metode for å fungere bra. Den er derimot testet med utsending av beskyttelsemeldinger og oppnår da et resultat på 200μs.

Sammensetting med 255 noder er testet på to måter. Mekanismen utløser utsending av topologipakker ved mottak av beskyttelsemeldinger i det ene forsøket og utløser ikke på det andre. Denne andre testen vil også gjenspeile situasjonen der det ikke blir mottatt noe beskyttelsemeldinger. Som forventet er snittet uten sending høyere. Det er derimot ikke stor forskjell på de to måtene å løse problemet på. Vi ser av figur 35 og 36 på neste side at resultatene har tydelige likhetstrekk.



Figur 35: Her har nodene utløst utsending av topologipakker ved mottak av beskyttelsemeldinger. Det er 255 noder på RPR-ringen. Den største verdien målt er ≈27,5ms, mens den minste er ≈3,2ms.

Forskjellen i starten er lett å se. Med utløsing av pakkesending er mange av resultatene samlet på et punkt. Dette er tiden det tar for topologipakkene å spre at det har skjedd en endring og at nodene har samlet inn den nye topologien. Uten utløsing vil den første puljen flyte utover. Dette kommer av at oppdateringen må bli utløst første gangen ved at en nodes topologitimer går ut. Forskjellene blant resultatene som kommer i den øverste puljen er minimale. At noen av resultatene kommer i en annen pulje med dårligere resultat avdekker en svakhet i denne mekanismen for nettverkskartlegging. Circa halvparten av resultatene kommer i denne andre puljen. Det er tilfeldig når en måling kommer i denne puljen. Ved sammensett-



Figur 36: Her har nodene ikke utløst utsending av topologipakker ved mottak av beskyttelsemeldinger. Det er 255 noder på RPR-ringen. Den største verdien målt er ≈27,5ms, mens den minste er ≈3,2ms.

ing av de to nettene blir det utløst veldig mange topologipakker for utsending i nodene. Alle topologipakkene som er på RPR-ringen fører til at pakkene blir forsinket på sin ferd rundt RPR-ringen. Det er derimot tilfeldig hvor mye den enkelte pakken blir forsinket på en runde. Det som skjer under circa halvparten av målingene er at den ene av de to topologipakkene en node sender ut blir mer forsinket enn den andre. Mellom mottak av de to pakkene blir pakkesending igjen utløst i noden som nullstiller variabelen “packetMissing”. Dette vil gjøre at de to først utsendte topologipakkene ikke kan brukes til å kalkulere et nytt topologibilde. Denne forsinkelsen i tiden topologipakken bruker rundt RPR-ringen vil skje flere ganger. Denne prosessen er veldig komplisert siden det er 255 noder som sender ut topologipakker til tilfeldige tider og utløst av forskjellige hendelser. Det er derfor vanskelig å vurdere hvorfor resultatene samler seg i en pulje rundt 25ms. Det er derimot vist med simuleringene at i de tilfellene resultatet kommer i den andre puljen så vil “packetMissing” nullstilles ofte.

Problemet nevnt over med nullstilling av “packetMissing” er også grunnen til at bryting av to linker oppfører seg annerledes med 255 noder enn 16 noder.

8.5 Har mekanismen oppfylt designkriteriene?

I denne delen vil hvert av designkriteriene som ble presentert i forrige kapittel (side 53) vurderes for å se om det er oppfylt.

Kriterie 1 Vi har sett av testene med ulike ringstørrelser at alle nodene oppnår et topologibilde. Kriteriet er dermed oppfylt.

Kriterie 2 Mekanismen følger Darwins forslag der nodene samler inn informasjon fra de andre nodene. Dette kriteriet er egentlig mer en avgjørelse på hvordan metoden skal løses.

Kriterie 3 Topologibildet kan sjekkes for konsistens. Dette gjøres ved sjekk av innkommende topologipakker mot bildet når status er OK. Kriteriet er dermed oppfylt.

Kriterie 4 Innholdet i topologipakkene er ikke drøftet i denne løsningen. Det er derimot sagt at nodene skal legge til sin MAC-adresse og MAC-tilleggsinformasjon. Det eksakte innholdet i MAC-tilleggsinformasjon er ikke fastsatt. Kriteriet er ikke fullstendig oppfylt.

Kriterie 5 Sirkulerende topologipakker på RPR-ringen vil merkes av mekanismen for nettverkskartlegging ved at konsistenssjekken utløses. De vil derimot ikke ødelegge eller påvirke et komplett topologibilde. Kriteriet er oppfylt.

Kriterie 6 Ved utsending av topologipakker settes TTL til maksimal ringstørrelse som er 255. Dette kommer av at pakkene må komme seg hele veien rundt RPR-ringen hvis denne inneholder 255 noder. En kan derimot se for seg situasjoner der mekanismen for nettverkskartlegging velger lavere verdier når den har fått et topologibilde.

Eksempel:
Hvis det er 16 noder på RPR-ringen kan TTL settes til 16. Tillegging av en node vil da føre til at pakken ikke kommer hele veien tilbake til avsender, men dette kan oppdages med en timer som går ut før pakkene returnerer. Hvis timeren går ut kan TTL økes før nye pakker sendes ut.

Dette er ikke testet ved simulering og det kan dermed ikke antas at dette er mere effektivt. Kriteriet er ikke bevist oppfylt.

Kriterie 7 Mekanismen sender ut topologipakker regelmessig og ved behov. Utløsing er også basert på koordinering med andre noder. Kriteriet er dermed oppfylt.

Kriterie 8 Dette punktet går mere på implementasjon enn de tidligere nevnte punktene. Vi har sett av testene at modellen kan legge til fjerne og endre informasjon i det lokale topologibildet. Kriteriet er dermed oppfylt.

Oppsummering

Denne metodens konsistenssjekk er antatt til å fungere bra. Det opprinnelige designet måtte imidlertid endres litt etter noen testsimuleringer ble kjørt. Dette gjelder i hovedsak oppførselen når RPR-ringen er wrappet. Resultatene for kjøringer med 16 og 255 noder er presentert i tabeller og drøftet. Oppførselen varierer på mange av testene ettersom det er 16 eller 255 noder på RPR-ringen.

9 Sammenligning av mekanismene for nettverkskartlegging

De tre forslagene som er presentert, utkastet sitt, Darwin og mitt eget har ulike egenskaper. Tidligere er de forskjellige presentert hver for seg. I dette kapittelet vil de vurderes sammen for å se hvilke deler av mekanismen som er like og forskjellige i de tre. På slutten av kapittelet vil også disse metodene sammenlignes kort med hvordan nettverkskartlegging gjøres i IEEE 1394.

Darwin kom som et forslag til utkast 0.1. Her var Darwin ett av tre muligheter som skulle vurderes. Når det neste utkastet kom hadde arbeidsgruppen bestemt seg for å velge en annet løsning på nettverkskartlegging. Etter hvert som utkastene har kommet ut i nye eksemplarer har utkastets metode for nettverkskartlegging også endret seg ut fra kommentarer fra arbeidsgruppens medlemmer. Utkast 1.0 var det fjerde i rekken og det tredje som inneholdt en metode for nettverkskartlegging. Det vil da være naturlig at Darwins metode ikke er like utviklet som utkastets metode. Dette på grunn av at Darwin ikke har gått gjennom den samme prosessen med forbedringer fra arbeidsgruppen. Min egen metode har blitt utviklet på grunnlag av den kunnskap jeg har fått fra å studere de fremlagte forslagene og simuleringer av disse. Min metode å løse problemer på har, som Darwin, ikke vært gjennom flere runder med arbeidsgruppen for feilretting og forbedringer. I et forslag som har blitt gjennomgått av mange personer vil det gjerne være mindre feil.

9.1 Testresultater

Denne delen vil sammenligne resultatene som har kommet fra de tre forslagene. Testene måler tider før alle noder har fått oppdatert sitt topologibilde. I alle de målte tilfellene vil så liten tid som mulig være ønskelig. Desto kortere tid man bruker på oppdatering, desto raskere vil nodene igjen kunne oppta kommunikasjon etter en feil på RPR-ringen. Bare mitt eget forslag er testet med 255 noder. Siden alle er testet med 16 noder vil resultater fra disse simuleringene bli brukt i sammenligningen av forslagene.

De ulike metodene blir sammenlignet for hver enkelt test. Dette er etterfulgt av en mer generell sammenligning.

9.1.1 Testene

Oppstart - test 1 Alle de tre får fullstendig oppdatert topologibilde i alle noder innen en rundetid. Dette kommer av at alle nodene vil begynne å sende topologipakker ved tid 0.

Fjernet node ved oppstart - test 2 Ved å fjerne en node midt under oppstartsprosessen vil oppdateringen til riktig topologibilde bli forsinket hos noen av nodene. Avhengig av hvilken metode som testes vil det kunne være noen noder uten topologibilde, noen med et topologibilde der den fjernede noden er med og andre med feil distanse til noen mottakernoder.

Utkastet får oppdatert sine topologibilder etter at alle nodene har sendt sine topologipakker på nytt. Dette vil ta den tiden man setter topologitimeren til å være. Min egen metode vil ha et resultat der oppdateringstidene ligner, men kan variere litt. En kan tenke seg at man starter to RPR-nett med hver sin mekanisme for nettverkskartlegging. Den aktuelle feilen påføres så

nettene. Alle nodene i nettet med utkast 1.0 mekanisme vil så sende ut sine pakker samtidig. Nodene i nettet med min egen metode vil derimot spre utsendingen av topologipakker på grunn av at den har en topologitimer som varierer rundt en fastsatt tid. Noen noder vil da kunne sende før andre og noen vil kunne sende etter nodene i nettet som har implementert utkast 1.0. I snitt så vil de derimot prestere likt.

Darwins forslag kommer dårligere ut. Denne vil kreve to timerperioder for at alle nodene skal bli oppdatert.

Fjerning av node - test 3 Ved fjerning av en node fra nettet vil nok en gang Darwin komme dårligst ut. Nodene er nødt til å vente til de samme topologipakkene har kommet to ganger før topologibildene kan oppdateres. Dette vil si at de må vente til alle nodene har sendt ut sine pakker to ganger. Ved perfekt spredning i utsending av topologipakker vil dette bety en tid på 2 timerperioder pluss en rundetid. Hvis derimot utsending av pakker ikke er spredt i det hele tatt (alle sender samtidig) vil snittiden for hvor lang tid det tar før alle har sendt sine pakker to ganger bli 1,5 timerperioder. Den halve er hvor lang tid det tar fra et tilfeldig punkt i tid til alle sender. Dette kan variere fra tilnærmet null tid til én timerperiode. Deretter må man vente én periode til.

Utkastet gjør oppdatering ved fjernet node etter første runde nettverkskartlegging. Dette tar da i snitt circa en halv timerperiode pluss en rundetid. I målingene ble disse resultatene jevnt fordelt fra null tid til én timerperiode.

Min egen metode presterer veldig bra i forhold til de andre metodene på dette punktet. Vi husker av figur 31 på side 66 at tiden før alle har oppdatert topologibilde er betydelig mindre enn timerperioden.

Lagt til node - test 4 Den tillagte noden vil under både utkastets og min egen metode sende ut topologipakker for å oppdatere sitt eget og de andres topologibilde. Dette vil i de to metodene føre til at også de andre nodene vil utløse pakkesending for å oppdatere poster og gi den nye noden informasjon om de som allerede finnes på ringen. Begge de to oppnår en tid på 1,5 rundetid. Noder på RPR-ringen som implementerer Darwin vil derimot ikke utløse pakkesending når de mottar pakker fra den nye noden. Dette fører også til at den nye noden ikke blir lagret i topologibildet før neste gang de andre nodene har sendt sine pakker. En er nødt til å vente på at topologitimeren i de forskjellige nodene skal gå ut.

Endring av identifikator - test 5 Endring av identifikator er et usikkert moment i utkast 0.3. I utkast 1.0 kommer det litt tydeligere frem hva som utløser pakkesending. Hvis en regner med at endring av identifikator også vil utløse pakkesending vil utkastets mekanisme prestere best av de tre med 1 rundetid. Min egen metode følger med 1,5 rundetid, mens Darwin må igjen vente to hele timerperioder. Utkastet er testet både med og uten pakkesending som gir to veldig forskjellige resultater.

Bryting av en link - test 6 Utkastet og min egen metode har sammenlignbare simuleringsresultater. Henholdsvis 300 og 340μs. Denne tiden må sees i sammenheng med størrelsen på RPR-ringen (hvilken rundetid den har). Grunnene til at det skiller 40 mikrosekunder på de

to forslagene kan være mange. Hovedgrunnen er derimot at på min egen metode må topologipakkene gå rundt hele ringen, også der de blir wrappet, før noe nytt topologibilde kan bygges opp. I utkast 0.3/1.0 blir det nye bildet bygget opp etterhvert som topologipakkene går langs ringene. De vil ikke bli wrappet og må derfor gå en kortere vei for å oppdatere topologibildene i alle nodene. Darwin har, som i de andre testene, ingen koordinert mekanisme og må vente på at timeren går ut to ganger.

Bryting av to linker - test 7 Min metode vil prestere dårligere enn utkastets med henholdsvis 230 og 70μs. for oppdatering.

Bryting av en og to linker vil føre til at nodene behandler topologipakker på en annen måte enn de ville gjort hvis ringen var feilfri. I utkastet vil ikke topologipakker videresendes hvis de kommer til en node som er wrappet. I min egen metode vil det foregå slik at det ikke legges til MAC-bindinger i topologipakkene når de passerer. Etter topologipakker har blitt wrappet vil heller ikke konsistenssjekken bli gjennomført. På grunn av dette vil tidene målt variere utifra hvilken metode man velger.

Sammensetting av brutt nett - test 8 Vi kan se for oss to situasjoner for utkastets og min egen metode. Én med utløsing av pakkesending og én uten. Uten pakkesending vil min metode prestere bedre enn utkastets. Min egen metode vil kunne ta nytte av den koordinerte mekanismen for å fortelle alle nodene at de må oppdatere sitt topologibilde, mens utkastet vil bli nødt til å vente på at timeren skal gå ut. Med pakkesending vil de to prestere veldig likt med 190 og 200μs. Darwin må igjen vente 2 runder.

9.1.2 Generelt

Vi ser at de forskjellige forslagene har noen likheter og flere forskjeller. Tidene for Darwin skiller seg mere ut fra de to andre enn de skiller seg fra hverandre. Utkastet har endel tester som gir bedre tider enn min egen metode, men i noen av testene er mine tider betydelig bedre. Det som er hovedforskjellen i min metode og utkastets er hva som utløser utsending av topologipakker. Utkastet er mer avhengig av beskyttelsemekanismen for at det skal bli utløst utsending, mens min egen metode kan raskere oppnå eget topologibilde uten hjelp fra andre mekanismer.

Det som også utgjør en stor forskjell på de tre forslagene er hvordan den koordinerte delen av mekanismen fungerer. Den koordinerte mekanismen for nettverkskartlegging vil føre til at det oftere, etter spesielle hendelser, blir sendt ut topologipakker fra nodene. Disse topologipakkene fører igjen til at topologibildene blir oppdatert.

Darwin har ingen koordinert mekanisme. Alle nodene er selv ansvarlig for å oppdatere sitt eget topologibilde uten hjelp fra de andre nodene. Utkastet og min egen metode bruker koordinert nettverkskartlegging i forskjellig grad. Utkastet utløser utsending av topologipakker basert på sjekk av innkommende topologipakker. Utifra hva man følger i utkastenes inkonsistente informasjon vil mekanismen variere i hvordan den presterer. Min egen metode har en konsistenssjekk som brukes på alle topologipakker i alle noder så fremt RPR-ringen ikke er wrappet.

9.2 Nettverkskartlegging i RPR versus IEEE 1394

I kapittel 3.3 på side 14 ble det presentert hvordan IEEE 1394 gjennomfører nettverkskartlegging. Her vil denne mekanismen sammenlignes med hvordan det gjøres i RPR.

Nettverkskartlegging i RPR og IEEE 1394 er ganske forskjellig. Dette kommer i hovedsak av at måten nettet i IEEE 1394 er bygget opp kan variere. I RPR er nodene koblet i en ring, mens nodene i IEEE 1394 kan kobles sammen vilkårlig. Det er derimot en viktig likhet mellom de ulike nettene. Nodene i IEEE 1394 sletter all informasjon i sitt topologibilde hvis det skjer en endring. Dette fører til at et nytt bilde må bygges opp. Informasjonen som finnes i det gamle bildet vil da ikke påvirke oppbyggingen av det nye topologibildet. I RPR har vi sett at å slette bildet kan være en fornuftig løsning for eksempel når en node skal fjernes fra topologibildet i utkastet (ref. kapittel 5.3.6.1).

Siden IEEE 1394 kan kobles sammen vilkårlig er det også mulighet for at det finnes ringer i dette nettets topologi. En av fasene i mekanismen for nettverkskartlegging har som oppgave å bygge opp en trestruktur av nettet som skjuler ringer i nettets fysiske topologi. RPR vil gjøre noe tilsvarende når det sendes data til en node kun langs den ene ringen.

Oppsummering

Dette kapittelet har sammenlignet resultatene fra testene kjørt på de forskjellige mekanismene. Vi har sett at Darwin skiller seg ut som den metoden som gir dårligst resultat. Utkastet og min egen metode har mange forskjeller men er også like på noen punkter.

10 Konklusjon og videre arbeid

Denne delen vil ta for seg oppgavens konklusjon. Den vil begynne med en generell oppsummering av rapporten. Kapittelet vil så drøfte om oppgavens problemstillinger og mål er løst før videre arbeid diskuteres.

10.1 Generelt

Vi har sett at kartlegging av nettets topologi er interessant både for utveksling av data og feiltoleranse. Nettverkskartlegging er nødvendig for å kunne sende data den korteste veien til mottakeren på RPR-ringen.

I en RPR-ring er topologien enkel. Alle noder, så fremt de ikke er broer, har to naboer. Nodene på RPR-ringen kan da deles i to grupper. De som kan nås langs den ytre ringen og de som kan nås langs den indre. Denne informasjonen, i tillegg til eventuell annen informasjon, blir lagret i en datastrukur som kalles topologibildet. Avstanden til hver enkelt node blir også lagret i topologibildet.

Det er mange hendelser som endrer RPR-ringens topologi. Bryting av linker, tillegging av nye noder og fjerning av noder er noen av de. Det er viktig at informasjonen i topologibildet så ofte som mulig er konsistent med RPR-ringens virkelige topologi. Etter endringer er det derfor ønskelig med en mekanisme som raskt klarer å oppdatere topologibildene i nodene på RPR-ringen.

I denne oppgaven er det presentert tre metoder å gjennomføre nettverkskartlegging på. De tre har forskjeller og likheter. Alle de tre finner nettets topologi ved de kjørte testene. Det er derimot stor differanse i tidene målt for hvor lang tid de ulike oppdateringene tar. Det som gjør at de målte tidene er så ulike er først og fremst koordinert nettverkskartlegging som bare utkastet og min egen metode har. Dette er meget viktig for at alle nodene raskt skal få vite om at det har skjedd en endring på RPR-ringen. Hvis en node, eller en annen mekanisme (for eksempel beskyttelse), på RPR-ringen oppdager at det har skjedd en endring kan nettverkskartlegging i alle nodene settes i gang. Dette innebærer at topologipakker sendes ut på begge ringene. Ettersom hvilken metode som brukes kan topologibildet bygges opp når pakkene har gått rundt RPR-ringen eller det bygges opp på veien rundt RPR-ringen i nodene som passeres.

10.2 Er problemstillingene løst?

Problemstillingene nevnt i kapittel 1.2 kan deles i to deler. Først er de grunnleggende problemstillingene nevnt. Blant disse seks blir det tatt opp blant annet hvordan en node vet hvilke andre noder som finnes på RPR-ringen og hvordan en node kan vite at en link mellom to noder har feilet. Disse seks problemstillingene er løst ved at en mekanisme for nettverkskartlegging opererer på RPR-ringen. De tre presenterte mekanismene har derimot ulik måte å løse de nevnte endringene i nettets topologi på.

Den andre delen tar for seg tre problemstillinger. Disse tre skiller seg fra de første seks ved at de vurderer hvordan de ulike mekanismene for nettverkskartlegging fungerer, mens de seks første tar for seg hvorfor nettverkskartlegging er nødvendig. Mekanismenes styrker og

svakheter er tatt for seg i flere deler av oppgaven. Mange svakheter er presentert rundt simuleringene da svakhetene har kommet opp under simuleringene av metodene. De to første av disse tre problemstillingene er dermed oppfylt. Den siste, som tar for seg hva forslagstillerne vektlegger, er derimot lite omtalt. Dette kommer av at forslagene er lagt frem på en slik måte at forslagsstillerne presenterer den aktuelle metoden, men diskuterer den ikke.

10.3 Er oppgavens mål løst?

Oppgavens mål er nevnt i kapittel 1.3 på side 3. Rapporten har tatt for seg presentasjon, evaluering og sammenligning av ulike forslag for nettverkskartlegging. Rapporten har gitt leseren kunnskap om forskjellige måter å løse nettverkskartlegging på og hva som kjennetegner de ulike metodene. Kjennskap til selve RPR teknologien er presentert som en innledning til oppgavens hovedproblemstilling, og leseren har dermed fått nødvendig kjennskap til RPR.

10.4 Videre arbeid

Hvis denne oppgaven skal brukes som et utgangspunkt for videre arbeid er det flere retninger man kan gå. Denne delen tar for seg noen retninger man kan gå med utgangspunkt i nettverkskartlegging. Omfanget av de ulike punktene varierer.

“Ringlet selection”, valg av ring

Valg av ring er en mekanisme nært knyttet til nettverkskartlegging. Valg av ring bruker blant annet topologibildet for å finne hvilken ring som raskest fører til hvilke noder. Forskjellige metoder for valg av ring kan variere i ytelse ettersom hvilken metode for nettverkskartlegging som velges. Det er derfor viktig å finne en god algoritme som velger ring samtidig ved utviklingen av en effektiv metode for nettverkskartlegging. Denne oppgaven har ikke tatt videre hensyn til valg av ring.

Oppsett av RPR-ringen

I denne oppgaven er mekanismene for nettverkskartlegging testet med et begrenset antall ringstørrelser og rundetider. En utvidelse av testene kan gjøres med flere oppsett av RPR-ringen. Dette vil gjøre at mekanismene blir enda grundigere testet. Hvilken arbeidsmengde dette punktet fører med seg vil avhenge mye av neste punkt som tar for seg simulatormodellens struktur.

Restrukturering av simulatormodellen

Under utviklingen av simulatormodellen var det flere problemer som oppstod. Hadde disse problemenes løsning vært kjent i starten av arbeidet kunne simulatormodellens struktur vært forbedret. Slik ville noe av arbeidet med simuleringene blitt forenklet og mekanismene kunne blitt testet grundigere.

Nettverkskartlegging ved sammenkobling av RPR-ringer

Som vist tidligere i oppgaven kan RPR-ringer kobles sammen. Kjennskap til noder utenfor RPR-ringen noden er tilknyttet til er nå ikke en del av nettverkskartlegging. En kan derimot tenke seg at fullstending kjennskap til tilkoblede RPR-ringer kan bli en del av nettverkskartlegging. Utvikling og testing av en slik mekanisme for nettverkskartlegging vil da være aktuelt. Dette er den av retningene over som fører med seg mest arbeid og kan være et utgangspunkt for en annen hovedfagsoppgave.

11 Ordbok

IEEE - Institute of Electrical and Electronics Engineers

RPR - Resilient Packet Ring, IEEE 802.17

LAN - Local Area Network - Betegner nettverk som spenner over korte avstander, gjerne innefor bygg eller nabobygg.

MAN - Metropolitan Area Network - Betegner nettverk som spenner over avstander innenfor byer og tilsvarende.

WAN - Wide Area Network - Betegner nettverk som spenner over store avstander.

Node - En datamaskin med tilkobling til et nettverk som kan sende og motta data på dette nettverket.

Ring - Et lukket sett av *noder* og *linker* i én retning slik at hver node har kun én link inn i noden og én link ut av noden[5]

RPR-ring - Et sett *noder* som tilsammen bygger opp et *RPR* nett ved hjelp av to *ringer* der dataoverføringen over ringene går motsatt vei.

Link - Den fysiske koblingen mellom *noder* for overføring av data.

Time-To-Live TTL - Et heltallsfelt i hver pakke som reduseres med 1 for hvert hopp pakker tar mellom noder i et nett

MAC lag - Delen av data link laget i osi-modellen som tar seg av tilgangen til det fysiske mediet.

Fysisk lag - Nederste lag i osi-modellen som tar seg av hvordan bit blir overført over en kommunikasjonskanal.

Rundetid - Summen av propagasjonstidene for alle *linkene* i en *ring*. Forkortet RTT.

Referanser

[1] Proposed Draft Standard. Part 17: Resilient packet ring access method and physical layer Specifications. Submitted to IEEE 802.17 as the Proposal - Gandalf. Draft 0.4 - November 8, 2001

http://grouper.ieee.org/groups/802/17/
(Draftet ligger på passordbeskyttede sider tilgjengelig kun for arbeidsgruppens medlemmer)

[2] Proposed Draft Standard. Part 17: Resilient packet ring access method and physical layer Specifications. Submitted to IEEE 802.17 as the proposal - Darwin. Draft 1.0 - January 29, 2002

http://grouper.ieee.org/groups/802/17/
(Draftet ligger på passordbeskyttede sider tilgjengelig kun for arbeidsgruppens medlemmer)

[3] Proposed Draft Standard. Part 17: Resilient packet ring access method and physical layer Specifications. Submitted to IEEE 802.17 as the DVJ Proposal. Draft 0.32:12, January 24, 2001

http://grouper.ieee.org/groups/802/17/
(Draftet ligger på passordbeskyttede sider tilgjengelig kun for arbeidsgruppens medlemmer)

[4] "Alladin" proposal presented in San Jose Updated Nov-09-01. Topology discovery proposal

[5] Part 17: Resilient Packet Ring Access Method & Physical Layer Specifications IEEE Draft P802.17/D0.3 June 10, 2002

http://grouper.ieee.org/groups/802/17/
(Draftet ligger på passordbeskyttede sider tilgjengelig kun for arbeidsgruppens medlemmer)

[6] Part 17: Resilient Packet Ring Access Method & Physical Layer Specifications IEEE Draft P802.17/D1.0 August 12, 2002

http://grouper.ieee.org/groups/802/17/
(Draftet ligger på passordbeskyttede sider tilgjengelig kun for arbeidsgruppens medlemmer)

[7] On Making TCP More Robust to Packet Reordering. Blanton / Allman 4/1 - 2002

[8] White Paper. Dynamic packet transport and technology.

http://www.cisco.com/warp/public/cc/techno/media/wan/sonet/dpt/dpta_wp.htm

[9] White Paper. Spatial reuse protocol

http://www.cisco.com/warp/public/cc/techno/wnty/dpty/tech/srpmc_wp.htm

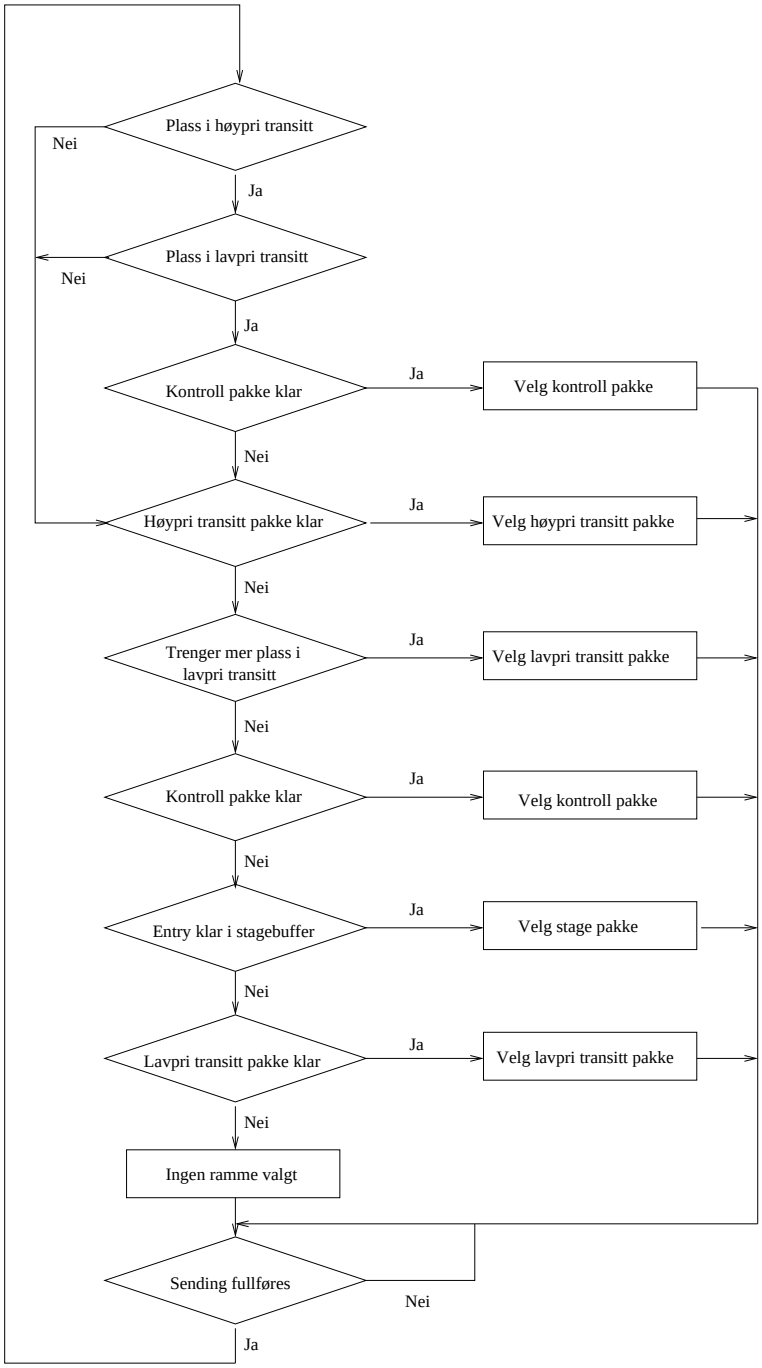
[10] Computer Networks - Network Topology
http://compnetworking.about.com/library/weekly/aa041601a.htm

[11] An Introduction to Resilient Packet Ring Technology
http://rpralliance.com/articles/ACF16.pdf

[12] Understanding Directory Services - Second Edition. Beth/Doug Sheresh

APPENDIX

A Figur 6.11 i kapittel 6.5.2 i utkast 0.3



Figur 37: Figuren viser tilstandsdiagram fra utkast 0.3. Diagrammet viser algoritmen som velger hvilket buffer pakker skal tas fra når det er klart for sending.