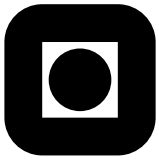
NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET

FAKULTET FOR INFORMASJONSTEKNOLOGI, MATEMATIKK OG ELEKTROTEKNIKK



HOVEDOPPGAVE

Kandidatenes navn: Stig Inge Lea Bjørnsen og Erik Vihovde Lohne

Fag: Datateknikk

Oppgavens tittel (norsk): Streaming i P2P-nettverk

Oppgavens tittel (engelsk): Streaming in P2P Networks

Oppgavens tekst:

Streaming av digitale medier har tradisjonelt vært gjort i klient/tjener-baserte systemer hvor en eller flere ressurssterke tjenere har levert data til et stort antall ressurskonsumerende klienter. P2P-systemer skalerer generelt bedre enn klient/tjener-systemer fordi samtlige deltakende noder bidrar med ressurser i tillegg til å konsumere dem. P2P er derfor et potensielt alternativ for mange tjenestetyper, deriblant streaming av digitale medier som video og lyd. P2P-arkitekturen introduserer imidlertid nye utfordringer og problemstillinger. Et viktig fokus i oppgaven er å gjøre en vurdering av de viktigste utfordringene knyttet til realisering av streamingtjenester i P2P-nettverk. Videre vil det være sentralt å identifisere hva som er de største hinderne i forhold til å kunne tilby tjenestekvalitet i slike systemer, samt å bidra med en selvstendig løsning for en eller flere av de viktigste problemstillingene.

Oppgaven gitt:	20. januar 2004
Besvarelsen leveres innen:	15. juni 2004
Besvarelsen levert:	15. juni 2004
Utført ved:	Institutt for datateknikk og informasjonsvitenskap
Veileder:	Roger Midtstraum

Trondheim, 15. juni 2004

Roger Midtstraum
Faglærer

Forord

Denne hovedoppgaven er utarbeidet som en del av sivilingeniørutdanningen i datateknikk ved Institutt for datateknikk og informasjonsvitenskap ved Norges teknisk-naturvitenskapelige universitet.

Formuleringen av oppgaveteksten har vært en kontinuerlig prosess, og utgangspunktet for hovedoppgaven samt den endelige oppgaveteksten er beskrevet i vedlegg A. Til denne hovedoppgaven følger også et separat vedlegg som inneholder kildekode og dokumentasjon til en implementert prototyp.

Hovedoppgaven er utført i løpet av vårsemesteret 2004 som et samarbeid mellom to avgangsstudenter ved gruppe for databaseteknikk.

Vi ønsker å takke vår veileder, førsteamanuensis Roger Midtstraum, som har vist oss stor tillit. Hans tilbakemeldinger og ideer har bidratt til at vi har opprettholdt en positiv framdrift gjennom hele semesteret. Vi takker også stipendiat Gisle Grimen for nyttige innspill.

Til sist ønsker vi å takke alle andre som har bidratt med innspill og velvillig stilt ressurser til rådighet.

Trondheim, 15. juni 2004

Erik Vihovde Lohne

Stig Inge Lea Bjørnsen

Sammendrag

Realisering av streamingtjenester i P2P-nettverk er en utfordring fordi slike nettverk er preget av dynamikk og stadige endringer i forhold til nodenes tilgjengelighet. Hvis en node som leverer en mediestrøm blir utilgjengelig, vil det oppstå avbrudd i avspillingen hos mottakernoden. Utgangspunktet for denne hovedoppgaven er et ønske om å kunne gjøre streaming med tilfredsstillende tjenestekvalitet til en realitet i P2P-nettverk. Målet er å spesifisere en metode som bidrar til å forbedre den opplevde ressurstilgjengeligheten i slike nettverk.

P2P-systemer kjennetegnes generelt ved at de består av noder på randen av Internett, at samtlige noder bidrar med ressurser og kommuniserer direkte med hverandre og at hver node er en selvstyrende enhet. Det finnes to hovedkategorier P2P-systemer, "ekte" og "hybride", samt en rekke applikasjonsområder. Streaming innebærer at innholdsleveranse gjøres i sanntid. Sanntidsleveranse krever at data leveres til mottaker i tide og at pakker leveres i rett rekkefølge. I P2P-nettverk, hvor omgivelsene preges av dynamikk og heterogenitet, er dette en stor utfordring. Flere teknikker kan benyttes for å håndtere slike omgivelser, men utfordringene knyttet til tilgjengelighet framstår som uløste og samtidig kritiske med tanke på oppnåelig tjenestekvalitet.

Vårt bidrag i retning av å forbedre den opplevde ressurstilgjengeligheten er en metode som blant annet benytter multinodestreaming som leveranseteknikk. Multinodestreaming innebærer å dele en strøm i flere delstrømmer og hente disse fra forskjellige noder. Delstrømmene kan da flettes ved mottak og spilles av som én strøm. Konsekvensen av at en node blir utilgjengelig vil kun bli kvalitetsdegradering av avspillingen, og ikke totalt avspillingsbrudd. Tilgjengeligheten kan forbedres ytterligere ved å utføre effektiv overtakelse av tapte delstrømmer. Ved å implementere metoden i en prototyp, er det mulig å teste dens funksjon og effekt. Testene viser at kvalitetsdegraderingen på avspillingen, som følge av at en node blir utilgjengelig, kun er midlertidig hvis en annen node kan overta leveransen. Reetablering av en delstrøm kan gjøres på gjennomsnittlig 4,7 sekunder. Enkeltnodestreaming ville til sammenligning medført fullt avspillingsbrudd. Resultatene sannsynliggjør derfor at multinodestreaming kan bidra til å forbedre den opplevde ressurstilgjengeligheten for streamingtjenester i P2P-nettverk.

Innhold

Innledning

Denne hovedoppgaven omhandler streaming av kontinuerlige medier i P2P-nettverk. Hovedoppgaven kan innholdsmessig deles i to. Den første delen består av utfyllende bakgrunnsmateriale for nærliggende temaer. Den andre delen er direkte relatert til problemstillingene som identifiseres i kapittel 1.

I kapittel 2 gjennomgås P2P-konseptet på en generell basis. Dette inkluderer definisjon, målsetninger, systemklassifikasjon og kartlegging av mulige applikasjonsområder. I tillegg vil også de viktigste teknologiske prinsippene som er felles for P2P-systemer bli forklart. Kapittel 3 gir en kort innføring i kontinuerlige medier. Det blir lagt vekt på å påpeke hvilke krav som kontinuerlige medier stiller til omgivelsene. Videre beskrives også kort hvilke teknikker som kan anvendes for å levere kontinuerlige medier over datanettverk, samt hvilke parametre som er viktige for å oppnå tilstrekkelig tjenestekvalitet for leveranse.

Kombinasjonen av P2P-nettverk og streaming av kontinuerlige medier gjennomgås i kapittel 4. Også her kartlegges potensielle applikasjonsområder, og deretter beskrives teknikker som kan brukes i systemer for streaming i P2P-nettverk. Videre identifiseres parametre som er knyttet til tjenestekvalitet for denne type applikasjoner.

En spesifikk metode for streaming i P2P-system, der en mottaker og flere avsendere samarbeider om å levere ett medieobjekt, spesifiseres og evalueres i kapittel 5. I kapittel 6 dokumenteres en prototyp som implementerer den spesifiserte metoden. Prototypen benyttes for å utføre ytelsestester, og resultatene av testen presenteres deretter. Videre blir den spesifiserte metoden, på bakgrunn av erfaringene fra prototypen, vurdert ut fra et overordnet perspektiv i kapittel 7. Dette innebærer blant annet diskusjon av parametre som påvirker ytelse og tjenestekvalitet. Tidligere uadresserte temaer knyttet til metoden for streaming i P2P-nettverk blir også berørt her.

Til slutt konkluderes hovedoppgaven i kapittel 8. I samme kapittel påpekes også mulighetene for videre arbeid som direkte eller indirekte er knyttet til metoden som ble spesifisert i denne hovedoppgaven.

Kapittel 1

Problemstilling

"Peer-to-Peer" (P2P) er en distribuert nettverksarkitektur som i dag benyttes for å realisere en rekke tjenester. Et av de vanligste P2P-baserte applikasjonsområdene er fildelingstjenester. Disse tjenestene muliggjør overføring av alle typer filer mellom to noder i et P2P-nettverk. Etter hvert som det har blitt vanlig å bruke datamaskiner til å spille av musikk og video, har det også blitt vanligere å distribuere multimedieobjekter i P2P-systemer. I vanlige P2P-systemer skjer nedlasting av slike objekter ved at hele objektet overføres over nettverket som en fil. Når hele filen er lastet ned, kan brukeren åpne den og spille av innholdet. Multimedieobjekter er ofte svært store, noe som gjør det tidkrevende å overføre dem over nettverk. Den tiden det tar fra en bruker starter mottak av et objekt til objektet kan tas i bruk, kaller vi ventetiden til objektet. Et av problemene i eksisterende P2P-systemer er at ventetiden for multimedieobjekter er svært stor.

"Streaming" er en teknikk som gjør det mulig å overføre et medieobjekt over nettverk på en slik måte at det kan spilles av hos mottaker før hele objektet er mottatt. Ved å benytte streaming for å levere et medieobjekt, kan ventetiden for objektet reduseres betydelig i forhold til om hele objektet måtte hentes i forkant av avspillingen. Ventetiden er derfor mer kritisk enn den totale nedlastningstiden. Siden medieobjekter som video og lyd har en tidsdimensjon, og avspillingen av objektet kun kan skje på ett punkt i tidsdimensjonen på et gitt tidspunkt, kan avspilling av slike objekter skje samtidig som de overføres over nettverket. Streaming i P2P-nettverk er imidlertid ikke bare enkelt, nettopp fordi det krever at sanntidsaspektene ved avspillingen ivaretas. Dette innebærer blant annet at data som sendes fra en avsendernode leveres hos mottakernoden i tide, men enda viktigere at det kontinuerlig blir levert data fra avsendernoden. Hvis mottakernoden blir sultefôret på data vil det kunne oppstå avspillingsbrudd.

Fordi mange P2P-nettverk består av vanlige og selvstyrende datamaskiner,

kan en node når som helst melde seg ut av nettverket eller bli utilgjengelig av andre årsaker. Denne egenskapen ved P2P-nettverk er en trussel i forhold til realiseringen av pålitelige sanntidstjenester, fordi brudd av dataleveransen vil medføre avspillingsstans. Et slikt avbrudd kan oppstå enten som følge av at en node ikke er i stand til å levere data med tilstrekkelig båndbredde, eller fordi avsendernoden ikke leverer data i det hele tatt. I begge tilfeller vil mottakernoden oppleve for lav datatilgjengelighet til at avspillingen kan skje kontinuerlig, og dermed også en tjeneste med dårlig eller ingen tjenestekvalitet (*eng. "Quality of Service", QoS).*

Utgangspunktet for denne hovedoppgaven er et ønske om å kunne gjøre streaming av digitale medier med tilfredsstillende tjenestekvalitet til en realitet P2P-nettverk. For å nå dette målet må blant annet utfordringene knyttet til tilgjengelighet løses. En streamingtjeneste med stor sannsynlighet for avspillingsavbrudd har ikke tilfredsstillende tjenestekvalitet. Skal streaming i P2P-nettverk bli et reelt alternativ til klient/tjener-baserte tjenester, må de tilfredsstille minstekrav til tjenestekvalitet. Fordi P2P-systemer ofte vil tjene andre formål enn for eksempel en multimedietjener, vil ikke nødvendigvis de samme kravene til tjenestekvalitet gjelde for både P2P- og klient/tjener-baserte systemer. Det er imidlertid nødvendig at et P2P-system som tilbyr streamingtjenester designes slik at ressurspotensialet i nettverket utnyttes fullt ut med det formål å tilby en pålitelig tjeneste. Tjenestekvalitet i form av tilfredsstillende tilgjengelighet er sannsynligvis en forutsetning for at P2P-baserte streamingtjenester kan være et alternativ til tjenester som er basert på klient/tjener-modellen.

I denne hovedoppgaven ønsker vi blant annet å kartlegge egenskaper knyttet til P2P-nettverk som påvirker muligheten til å realisere streaming av digitale medier med tilfredsstillende tjenestekvalitet. Ideen om å utføre streaming av digitale medier i en P2P-basert arkitektur er ikke ny. Eksisterende forskning har adressert flere aktuelle problemstillinger som er direkte knyttet til streaming i P2P-nettverk. Det eksisterer imidlertid lite forskning som omhandler tilgjengelighetsaspektet for slike systemer. Mange av teknikkene som har blitt foreslått for å løse andre problemstillinger kan likevel ha anvendelsesområder i forhold til de problemstillingene som er fokus i denne hovedoppgaven. En naturlig introduksjon til å analysere og løse problemstillingene knyttet til tilgjengelighet er derfor å presentere og vurdere eksisterende forskningsmateriale, og spesielt resultater som kan spille en rolle i forhold til tjenestekvalitet.

Et viktig resultatmål for hovedoppgaven er en spesifikasjon av en metode som kan bidra til å øke tilgjengeligheten (slik den oppleves av sluttbrukeren) i P2P-nettverk som tilbyr streamingtjenester. Dette ønsker vi å gjøre ved å identifisere hindere knyttet til tilgjengelighet, og foreslå en løsning som bidrar til å redusere virkningen av disse. Det er ønskelig at datatilgjengeligheten, slik den oppleves av brukeren, kan forbedres ved å utnytte det eksisterende ressurspotensialet i P2P-nettverket, og ikke ved endre egenskapene til selve systemet.

Ved å implementere den spesifiserte metoden i en prototyp som opererer i et reelt P2P-nettverk, ønsker vi å kunne vurdere metodens funksjon og effektivitet. Målet med prototypen er å bevise at metoden bidrar til forbedret tilgjengelighet uten at det må gjøres store endringer i forhold til omgivelsene til P2P-systemet, eller at det må stilles urimelige krav til hver enkelt node. Som en effekt av disse resultatene og vårt arbeid generelt, ønsker vi å bidra til at arbeidet med å realisere pålitelige streamingtjenester i P2P-nettverk kan bringes et skritt videre.

Kapittel 2

P2P-systemer

"Peer-to-Peer" som begrep har de senere år blitt sterkt opphauset, og resultatet er at begrepet også brukes for å beskrive systemer som i grunnen ikke baseres på P2P-prinsipper. For å oppklare hva som egentlig menes med P2P er det derfor nyttig å ha en definisjon som påpeker de ulike elementene som kjennetegner P2P-systemer. Utover en definisjon er det også behov for å kunne klassifisere P2P-systemer både etter hvilke teknologiske prinsipper de er basert på, samt etter hvilke applikasjonsområder de egner seg for. Selve grunnideen for P2P om å bygge systemer bestående av likeverdige noder er i seg selv ikke ny, men moderne P2P-systemer baseres i tillegg på nye teknikker. Det er derfor viktig å påpeke hvordan nyere P2P-teknologi kan integreres med eksisterende kommunikasjonsteknologi.

2.1 Definisjon

Det er gjort flere forsøk på å definere begrepet "Peer-to-Peer". Mange av definisjonene har likheter, men det har foreløpig ikke vært enighet om én definisjon å forholde seg til.

Ordet "peer" er engelsk og stammer opprinnelig fra det latinske ordet "par" som betyr likeverdig. Et P2P-nettverk vil ut fra denne betydningen være et nettverk av likeverdige deltakere, og samtlige noder vil da spille de samme rollene. Dette er også tilfelle i mange P2P-nettverk hvor det ikke finnes en sentral tjener og hvor nodene kun kommuniserer med hverandre. Det er likevel også andre karakteristikker som kjennetegner P2P-nettverk.

Rüdger Schollmeier definerer P2P på følgende måte [Schollmeier, 2002]:

> "En distribuert nettverksarkitektur kan kalles et P2P-nettverk hvis deltakerne deler en del av deres egne maskinvareressurser (proses-

> sorkraft, lagringskapasitet, nettverkskapasitet, printere, ...). Disse delte ressursene er nødvendige for å tilby tjenestene og innholdet som tilbys av nettverket (for eksempel fildeling eller delte arbeidsområder for samarbeid). De er direkte aksesserbare for andre noder uten å måtte gå via mellomliggende entiteter. Deltakerne i et slikt nettverk er dermed ressursinnskytere så vel som ressurskonsumenter" (oversatt fra engelsk).

Denne definisjonen fremhever likeverdighetsprinsippet ved at alle noder både skal tilby og konsumere ressurser (med andre ord både fungere som tjener og klient). I tillegg spesifiserer Schollmeier at ressursene som de enkelte deltakerne tilbyr er nødvendige for at tjenester og innhold skal være tilgjengelige i nettverket.

Clay Shirky bruker følgende definisjon/beskrivelse [Shirky, 2001]:

> "P2P er en klasse applikasjoner som utnytter ressurser (...) på randen av Internett. Fordi det å aksessere disse desentraliserte ressursene medfører at man må operere i omgivelser med ustabil tilkoblingsbarhet, må P2P-noder operere utenfor DNS-systemet og ha betydelig eller totalt selvstyre i forhold til sentrale tjenere" (oversatt fra engelsk).

Det er altså noder på randen av Internett, tilsvarende klient-noder i et klient/tjener-nettverk, som utgjør ressursene i P2P-nettverk. Et P2P-nettverk må ha et eget navnesystem som er verken er avhengig av sentrale tjenere eller lokasjon/adressering i de underliggende nettverkslagene. En node må når som helst kunne melde seg inn eller ut i et P2P-nettverk, noe som gjør at nettverket stadig er i endring og må tilpasse seg dynamisk etter omgivelsene. I tillegg må de deltakende nodene kunne delta i nettverket uten en sentral styringsenhet.

Det finnes også mange andre definisjoner som ikke nevnes her. Noen likner på de definisjonene som allerede har blitt nevnt, mens andre har et noe annet fokus eller er mer applikasjonsspesifikke. De karakteristikkene som går igjen i de ulike definisjonene gir et godt bilde av hva som legges til grunn for å kunne kalle et system for P2P. For resten av hovedoppgaven legges følgende hovedkriterier til grunn for at et system skal klassifiseres som P2P:

1. Noder i et P2P-nettverk befinner seg på randen av internett.
2. Samtlige noder kan både bidra med og konsumere ressurser.
3. Et P2P-nettverk har et adresseringssystem som er uavhengig av DNS.
4. Noder kan kommunisere direkte med hverandre.

5. Et P2P-nettverk må kunne operere i omgivelser med ustabil tilkoblingsbarhet (*eng. "connectivity"*).

6. Noder i et P2P-nettverk må være selvstyrende enheter.

Det finnes systemer som bare delvis oppfyller disse kriteriene men som likevel klassifiseres som P2P. Eksempelvis finnes det P2P-systemer som delvis er avhengig av en sentral tjener for å ivareta noen funksjoner, men som ellers har karakteristikker som P2P. Klassifiseringskriteriene fungerer likevel som en viktig indikator på hvilke systemer som er P2P og hvilke som ikke er det.

2.2 Mål med P2P

P2P kan lett forveksles med andre begreper, slik som tradisjonell distribuert databehandling og ad hoc-nettverk. Riktignok kan P2P både brukes til distribuert databehandling og ha ad hoc karakteristikker, men det er andre egenskaper som ofte legges til grunn for P2P-begrepet. Med bakgrunn i definisjonene av P2P, vil dette delkapittelet gå et skritt videre i retning av å klassifisere P2P ved å se på hvilke mål som ofte ligger til grunn for valg av en P2P-arkitektur.

Kostnadsdeling I klient/tjener-systemer med mange klienter er det som regel tjeneren som bærer kostnaden ved å tilby en tjeneste. Jo flere klienter det er i systemet, desto dyrere blir tjeneren. P2P-systemer har som mål å dele kostnaden mellom de ulike nodene i nettverket ved at hver node bidrar med en andel av de nødvendige ressursene for å kunne tilby en tjeneste. Slike ressurser kan for eksempel være lagringskapasitet, prosessorkraft eller nettverksbåndbredde.

Forbedret skalerbarhet Skalerbarhet er et av de viktigste målene med P2P-systemer. Den største flaskehalsene i klient/tjener-arkitekturen er tjeneren. Jo flere klienter, desto større blir ressurskravet som tjeneren må tilfredsstille. P2P-systemer løser dette skalerbarhetsproblemet ved at det ikke finnes sentrale enheter og at hver node har stor grad av selvstyre. I tillegg bidrar hver node med ressurser, noe som gjør at den totale mengden tilgjengelige ressurser i systemet øker i takt med antall ressurskonsumenter.

Aggregering av ressurser Det at hver node i et P2P-nettverk bidrar med ressurser gjør også at det er mulig å aggregere ressurser for å utføre større oppgaver eller tilby ressurskrevende tjenester. Eksempler på slike oppgaver er utregning av større matematiske problemer og lagring av store datamengder. Hver node i nettverket bidrar med en liten andel ressurser som til sammen er tilstrekkelig for å utføre en oppgave.

Økt selvstyre I mange tilfeller er det ikke ønskelig å bruke en sentralisert enhet for å utføre en oppgave. Det kan være mange årsaker til dette, for eksempel for å unngå at én node i nettverket (tjeneren) får ansvaret for oppgavene som gjøres. Ved å spre ansvaret til selvstendige noder, kan systemet bli mindre sårbart.

Fleksibilitet P2P-systemer støtter i stor grad fleksibilitet ved at de implementerer et eget logisk nettverkslag på toppen av eksisterende infrastruktur for å støtte ruting i dynamiske omgivelser. Dette tillater "ad hoc kommunikasjon" hvor noder kan melde seg inn og ut samtidig som systemet opprettholder sin funksjon. Dette er blant annet ønskelig i systemer som utnytter ledige maskinressurser på randen av internett, for eksempel SETI@Home [Seti@home, 2004].

Også andre funksjonalitetsmål kan lede mot valget av en P2P-arkitektur. Målene som er nevnt her er valgt fordi de gir en god indikasjon på hva som kan være bakgrunnen for valget av P2P, samt hvilken funksjonalitet P2P kan implementere.

2.3 P2P i et historisk perspektiv

P2P er ikke en ny kommunikasjonsarkitektur på Internett. Faktisk var Internett opprinnelig bygd på P2P-lignende kommunikasjonsmønstre. Andre arkitekturer har vært dominerende i nyere tid, men P2P er nå på vei tilbake. Dette delkapittelet beskriver P2P i et historisk perspektiv i lys av hvordan Internett generelt har utviklet seg fra 1960-tallet og fram til i dag.

2.3.1 Den opprinnelige ideen med Internett (1969)

Internett har sin opprinnelse i ARPANET fra 1969. Målet med ARPANET var å dele prosesseringsressurser i USA ved å integrere ulike systemer i ett stort nettverk hvor alle noder var likeverdige deltakere. Samtlige noder kunne altså både bidra med og konsumere ressurser, og kommuniserte i så måte på P2P-vis. P2P er altså ingen ny idé, men tvert imot den opprinnelige ideen med Internett [Minar og Hedlund, 2001].

De mest brukte tjenestene i det tidlige Internett var tjenester som FTP og Telnet. Disse tjenestene var i seg selv klient/tjener-applikasjoner. En FTP-klient logget seg på en FTP-tjener og lastet filer opp eller ned. Denne funksjonaliteten var imidlertid ikke én-veis, for alle noder i nettverket kunne være en FTP-tjener for hvilken som helst annen node i nettverket. Kommunikasjonsmønstrene var

altså symmetriske. En annen viktig karakteristikk ved det tidlige Internett var at de fleste noder hadde statiske IP-adresser.

Etter hvert som Internett ble tatt mer og mer i bruk oppstod nye tjenester, som for eksempel DNS. DNS er et eksempel på et system som kombinerer P2P-teknologi med en hierarkisk modell for informasjonsoppslag. DNS knytter tekstbaserte navn opp mot IP-adresser, og var opprinnelig designet for å støtte noen få tusen oppslag. DNS har imidlertid har vist seg å være i stand til å skalere flere tusen ganger [Minar og Hedlund, 2001].

2.3.2 "World Wide Web" (1994)

1994 regnes av mange som et vendepunkt for utviklingen av Internett. Forbrukermarkedet for Internett-tilgang nærmest eksploderte, og "World Wide Web" (WWW) ble en av de mest brukte tjenestene [Cailliau, 2000]. Dette fikk store konsekvenser for bruksmønstrene på Internett. Private brukere hadde ikke råd til permanent oppkobling mot Internett, noe som gjorde at vanlige analoge telefonlinjer og modem ble brukt for kommunikasjon med operatørnettet. Disse forholdsvis dårlige Internett-forbindelsene, sammen med endret bruksmønster, gjorde at klient/tjener-kommunikasjon ble mer og mer vanlig og tok over for den tidligere P2P-kommunikasjonen. Private datamaskiner var rett og slett ikke i stand til å kommunisere i P2P-nettverk fordi de ikke hadde tilstrekkelig nettverkskapasitet for å tilby tjenester til andre.

Den nye utviklingen gav opphav til et operatørnett som vanskeliggjør P2P-kommunikasjon. De vanligste tjenestene, hvor sluttbrukere stort sett er tjeneste*konsumenter*, bidro til et asymmetrisk bruksmønster. Dette bidro videre til at kommunikasjonskanalen mellom brukermaskiner og operatørnettet ble asymmetrisk. "Moderne" Internett-forbindelser, som ADSL eller Internett over eksisterende infrastruktur opprinnelig beregnet for kabel-TV, har tilpasset seg klient/tjener-omgivelser og har derfor asymmetrisk båndbredde. Som regel tillater slike forbindelser flere ganger større båndbredde fra Internett til brukermaskin i forhold til den motsatte veien. Når datamengden som passerer mellom Internett og en brukernode er asymmetrisk, er det også mest effektivt å ha asymmetrisk båndbredde. P2P-kommunikasjon, som blant annet kjennetegnes ved at samtlige noder er likeverdige og dermed både kan fungere som klient og tjener, er mindre egnet i asymmetriske nettverk [Minar og Hedlund, 2001].

Også andre faktorer har bidratt til å vanskeliggjøre P2P-kommunikasjon i dagens operatørnett. På grunn av den økende faren for datainnbrudd fra andre brukere, har brannmurer blitt vanlige. Brannmurer er imidlertid et problem for P2P-kommunikasjon fordi deler av nettverket blir utilgjengelig for andre noder. Dynamiske IP-adresser har også blitt vanlig fordi mange av nodene i nettverket

er avslått eller frakoblet i lengre perioder. Statiske IP-adresser ville medført at mange IP-adresser var opptatt selv om de ikke er i bruk. Med et begrenset antall IP-adresser, har de fleste operatører valgt å tildele disse adressene dynamisk slik at et stort antall noder deler på et mindre antall IP-adresser. Stadige adresseendringer gjør det vanskelig for en node å fungere som vertsmaskin for en tjeneste, noe som dermed vanskeliggjør P2P-kommunikasjon. En annen vanlig teknikk for å "spare" på de globale IP-adressene er bruken av "Network Address Translation" (NAT), som oversetter et sett av globale IP-adresser til et større sett av lokale IP-adresser. NAT muliggjør at flere noder kan dele global IP-adresse, noe som i praksis gjør at en node ikke kan adresseres ved hjelp av bare IP-adressen. NAT har vist seg å være en stor utfordring for P2P-systemer [Minar og Hedlund, 2001].

2.3.3 Nye P2P-applikasjoner (2000–)

Som beskrevet innledningsvis var Internett opprinnelig designet for P2P-kommunikasjon. Etter hvert som Internett utviklet seg på 1990-tallet, ble klient/tjener-arkitektur mer og mer vanlig. Denne arkitekturen er godt egnet til sentralisert fillagring, søkemotorer og for å "surfe på nettet".

Mot slutten av 1990-tallet endret vilkårene for kommunikasjon på Internett seg dramatisk. Lagrings- og prosesseringskapasiteten til vanlige datamaskiner utviklet seg markant på få år, og operativsystemene var blitt langt mer sofistikerte. Denne utviklingen gjorde at klientmaskiner kunne spille en langt mer aktiv rolle på Internett. Som et resultat av dette begynte nye P2P-applikasjoner å se dagens lys. Napster [napster.com, 2004] er et eksempel på en slik tjeneste. Napster gjorde at vanlige Internettbrukere kunne dele musikkfiler mellom seg ved hjelp av Internett. Mange andre P2P-baserte applikasjoner oppstod i årene etter, og blant annet fildeling ble et populært applikasjonsområde.

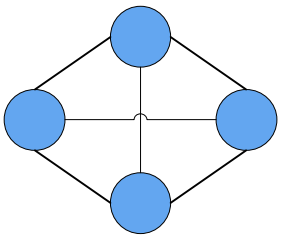
En av de største begrensningene for P2P-kommunikasjon i dag er Internett-forbindelser med asymmetrisk båndbredde. For P2P-applikasjoner, hvor hver node også må tilby nettverksressurser i tillegg til å konsumere dem, begrenser dette P2P-systemets tilgjengelige ressurser. Dynamiske IP-adresser, brannmurer og NAT kan fortsatt skape problemer, men det finnes løsninger som omgår disse hindrene.

Dynamiske IP-adresser og NAT har oppstått som følge av mangel på globale IP-adresser. IP versjon 4 (IPv4) har et adresserom på 32 bit. Med IP versjon 6 (IPv6) vil adresserommet utvides til 128 bit, noe som tillater $2^{128-32} = 2^{96}$ ganger så mange adresser som dagens IPv4 [Deering og Hinden, 1998]. IPv6 vil føre til at dynamiske IP-adresser og NAT blir unødvendig med mindre de tjener et annet formål enn rasjonering av adresser.

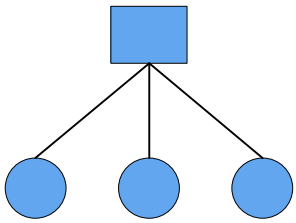
Trendene på private Internett-oppkoblinger går mot stadig høyere båndbredde

både inn og ut. Flere Internettoperatører tilbyr i dag private Internettforbindelser med symmetrisk båndbredde på opptil 10 Mb/s [Lyse AS, 2004]. Med denne trenden kan P2P-applikasjoner gå en lysere tid i møte, og mange nye applikasjonsklasser vil kunne oppstå.

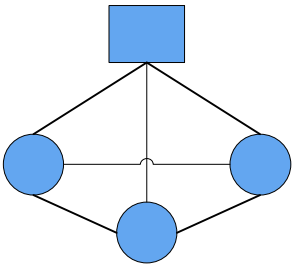
Figur 2.1 oppsummerer hvordan kommunikasjonstrendene på Internett har endret seg gjennom tidene. I figuren er tjenere representert som rektangler, mens klienter og P2P-noder er representert som sirkler. Den første perioden, illustrert i figur 2.1(a), var sterkt dominert av direkte kommunikasjon mellom likeverdige noder (P2P). Den siste halvdelen av 1990-tallet, hvor datamaskiner med dårlige nettverksressurser begynte å ta i bruk Internett, er illustrert i figur 2.1(b). Trenden i denne perioden var sterkt dominert av klient/tjener-kommunikasjon. Figur 2.1(c) viser trenden i dagens Internett. Klient/tjener-arkitekturen har bestått for mange tjenester, mens P2P-arkitektur har blitt mer og mer vanlig for mange andre tjenester, deriblant fildeling.



(a) 1969–1995: P2P



(b) 1995–2000: Klient/tjener



(c) 2000–2004: Kombinasjon

Figur 2.1: Internett-tjenester — trender i kommunikasjonsarkitektur.

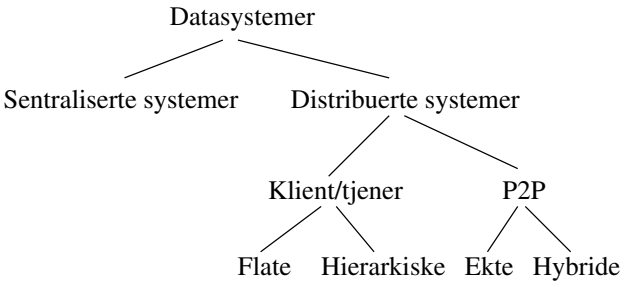
2.4 Klassifikasjon

P2P er en fellesbetegnelse for systemer som kan ha vidt forskjellige egenskaper, og det er derfor nyttig å kunne klassifisere P2P-systemer ytterligere etter system- og applikasjonskategorier.

2.4.1 Systemklassifikasjon

Figur 2.2 illustrerer hvordan P2P-systemer kan klassifiseres i forhold til øvrige datasystemer. Av definisjonen (gitt i kapittel 2.1) går det fram at P2P-baserte systemer er distribuerte, og de er derfor plassert som en underkategori av "Distribuerte systemer". En av de viktigste særtrekkene ved P2P-systemer er at alle nodene har mulighet til både å bidra med og forbruke samtlige typer ressurser. P2P-modellen avviker derfor fra andre modeller for distribuerte systemer med tanke på likeverd mellom nodene, men også ved nodenes grad av selvstyre. Etablerte modeller for distribuerte systemer forutsetter at systemet må administreres, og i motsetning til P2P-systemer utfører også de deltakende nodene forskjellige oppgaver.

I figur 2.2 inngår både tradisjonelle klient/tjener-baserte systemer (WWW) og systemer basert på distribuerte komponenter (som for eksempel CORBA [OMG, Inc., 2002] og DCOM [Microsoft Corp., 1996]) under kategorien "Klient/tjener".



Figur 2.2: Klassifikasjon av datasystemer [Milojicic et al., 2002].

2.4.2 P2P-klassifikasjon

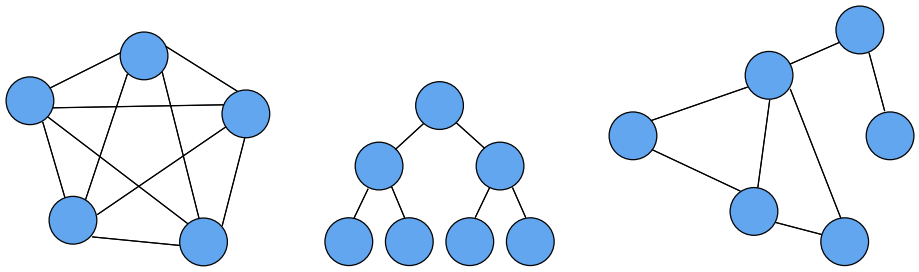
P2P-systemer klassifiseres etter grad av desentralisering og etter hvilken nettverkstopologi de benytter seg av. P2P-systemer uten noen form for sentrale enheter kalles *desentraliserte* eller *ekte*. P2P-systemer som er avhengige av en eller flere sentraliserte enheter kalles *delvis sentraliserte* eller *hybride* [Melville et al., 2002].

Desentraliserte (ekte) P2P-systemer

Et ekte P2P-system kjennetegnes ved at det er fullt operativt uten noen form for sentralisert ressurs. Dette innebærer at enhver deltaker i nettverket kan fjernes uten at noen form for funksjonalitet går tapt. Ekte P2P-systemer har blant annet blitt beskrevet som følger [Schollmeier, 2001]:

> "Alle deltakere har samme rettigheter og plikter, selv om de avhenger av ulik maskinvare. Følgelig eksisterer det i "Peer-to-Peer"-nettverket ingen sentral enhet for å administrere, koordinere eller tilby innhold/tjenester. Derfor unngår nettverket ethvert kritisk punkt (*eng. "single-point-of-failure"*), noe som gjør tjenestenektsangrep (*eng. "Denial of Service", DoS*) tilnærmet umulige, siden alle oppgaver distribueres utover nettverket" (oversatt fra engelsk).

Figur 2.3 viser tre nettverkstopologier som kan benyttes av ekte P2P-systemer. Nettverkstopologiene viser, i tilfellene for indirekte kommunikasjon, ulike måter for hvordan meldinger mellom noder kan utveksles via mellomliggende noder før de involverte nodene har kjennskap til hverandre. Etter at to noder indirekte har kontaktet hverandre og utvekslet meldinger, kan de kommunisere direkte med hverandre.



(a) Direkte kommunikasjon.

(b) Strukturert indirekte kommunikasjon.

(c) Ustrukturert indirekte kommunikasjon.

Figur 2.3: Desentraliserte P2P-topologier [Melville et al., 2002].

Direkte kommunikasjon En nettverkstopologi basert på direkte kommunikasjon (figur 2.3(a)) innebærer at samtlige noder kjenner til hverandres eksistens. I P2P-systemer med veldig mange deltakende noder vil dette stille krav til hver nodes minne- og prosessorkapasitet. Direkte kommunikasjon medfører

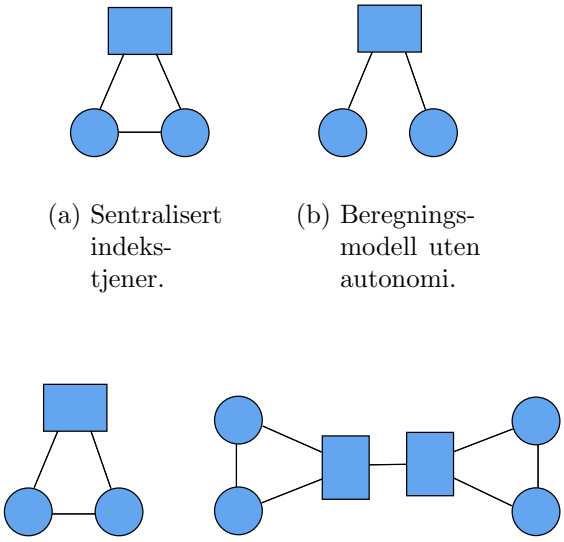
at samtlige noder må få beskjed når en node melder seg inn eller ut av nettverket. Siden P2P-systemer er flyktige med noder som stadig melder seg inn og ut, vil en slik topologi derfor kreve at hver node har tilstrekkelig båndbredde til å kunne utveksle disse meldingene.

Strukturert indirekte kommunikasjon Indirekte kommunikasjon innebærer at en node kun har kjennskap til sine egne nabonoder. Hver node må derfor kontakte andre noder via sine egne nabonoder. Indirekte kommunikasjon unngår dermed problemet med at hver node må varsles når en annen node melder seg inn eller ut av P2P-systemet. Nettverket er organisert etter en fast struktur, og fordelen med dette er at hver node vil ha et begrenset antall indirekte kommunikasjonsledd for å nå enhver annen node. Opprettholdelse av strukturen krever at nodene er i stand til å utføre en form for samordnet organisering. Figur 2.3(b) viser en hierarkisk struktur, men andre mulige topologier er for eksempel ring eller stjerne.

Ustrukturert indirekte kommunikasjon Ustrukturert indirekte kommunikasjon innebærer at hver node har kjennskap til et begrenset antall nabonoder. Kommunikasjon med ukjente noder skjer også her indirekte via de kjente nabonodene. Siden P2P-nettverket ikke har en fast struktur er det ikke mulig å gi en øvre grense for antall indirekte ledd ved kommunikasjon mellom to noder. Den ustrukturerte topologien krever ingen samordning mellom nodene, og konsekvensen av at en node melder seg inn eller ut blir dermed mindre enn hva som er tilfelle for strukturert indirekte kommunikasjon.

Delvis sentraliserte (hybride) P2P-system

P2P-systemer med minst en sentral enhet som utfører en kritisk oppgave i nettverket klassifiseres som delvis sentraliserte eller hybride P2P-systemer. Eksempler på oppgaver som kan utføres av en sentral enhet er lokalisering av ressurser, kontroll av nodene i nettverket og fordeling av oppgaver til deltakende noder.



(a) Sentralisert indeks-tjener.

(b) Beregnings-modell uten autonomi.

(c) Beregnings-modell med autonomi.

(d) Flere tjenere.

Figur 2.4: Delvis sentraliserte P2P-topologier [Melville et al., 2002].

Sentralisert indekstjener En sentralisert indekstjener i et hybrid P2P-system (figur 2.4(a)) tilbyr lokalisering av ressurser. Et slikt system vil ha en klient/tjener-lignende relasjon mellom deltakende noder og den sentrale indekstjeneren. Når en node har kontaktet indekstjeneren og mottatt lokasjonen til en ønsket ressurs, foregår all videre kommunikasjon direkte mellom de deltakende nodene. Deltakende noder både bidrar og forbruker ressurser, og et slikt system klassifiseres derfor som et P2P-system.

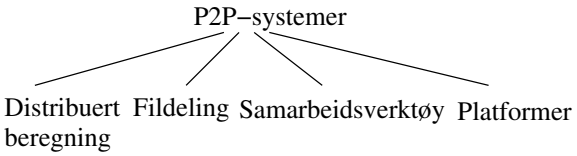
Beregningsmodell uten autonomi For å utføre tyngre beregninger kan prosessorkraften i deltakende noder utnyttes. Figur 2.4(b) viser en modell for distribuerte beregninger der all kontroll utføres av en sentral tjener. Den sentrale tjeneren deler opp og fordeler oppgaver mellom de deltakende nodene. I tillegg vil tjeneren også være ansvarlig for å videreformidle all kommunikasjon mellom de deltakende nodene. Den sentrale tjeneren har dermed full kontroll over samtlige noder. Strengt tatt kan ikke et slikt system klassifiseres som et P2P-system, men på grunnlag av at systemer basert på denne modellen allerede er tatt i vidstrakt bruk, nevnes modellen ofte i P2P-litteraturen [Melville et al., 2002].

Beregningsmodell med autonomi En beregningsmodell med autonomi (figur 2.4(c)) har også et sentralt kontrollpunkt i form av en tjener, men tillater at hver deltakende node i større grad styrer seg selv. I motsetning til beregningsmodellen uten autonomi kan nodene kommunisere direkte med hverandre. Dette muliggjør at nodene kan sende forespørsler til hverandre i tillegg til den sentrale tjeneren. Selv om nodene nå står friere til å sende ut forespørsler er det fortsatt det sentrale kontrollpunktet som tar initiativet når en ny oppgave skal utføres av systemet.

Flere tjenere Ved å introdusere flere sentrale tjenere i et hybrid P2P-system minker sannsynligheten for at systemet blir utilgjengelig. Gruppen av sentrale tjenere kan enten framstå som en stor virtuell tjener eller som et sett av mindre separate tjenere. En interessant egenskap ved slike systemer er at de kan benytte seg av ulike P2P-modeller i hvert av nivåene. Deltakende noder kan for eksempel benytte seg av sentrale indekstjenere, mens kommunikasjon mellom de sentrale indekstjenerne kan baseres på direkte kommunikasjon.

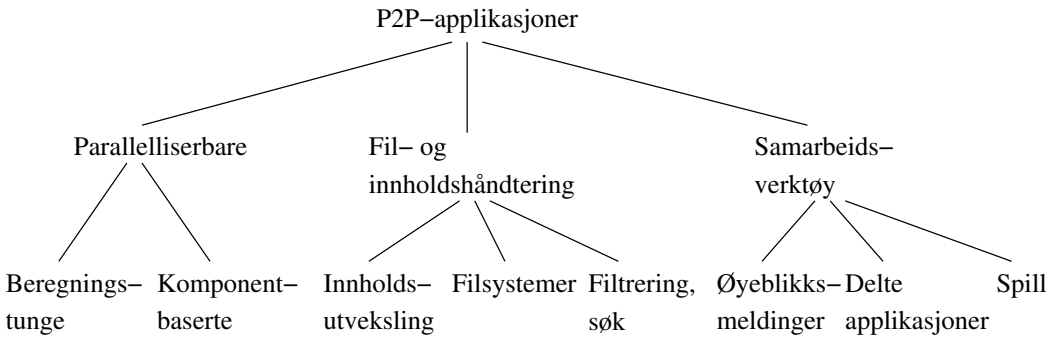
2.5 Applikasjonsområder

P2P-systemer kan også klassifiseres etter hvilke applikasjonsområder de retter seg mot. De overordnede applikasjonsområdene der P2P-teknologi anvendes er distribuerte beregninger, fildeling og samarbeidsverktøy (figur 2.5). Enkelte P2P-systemer, som for eksempel JXTA [Project JXTA, 2004], fungerer i tillegg som mellomvare for P2P-applikasjoner, og disse havner dermed i en fjerde systemkategori, plattformer [Milojicic et al., 2002]. Klassifikasjon av P2P-systemer etter applikasjonsområde er uavhengig av om de er ekte eller hybride.



Figur 2.5: Klassifikasjon av P2P-systemer [Milojicic et al., 2002].

Figur 2.6 viser de ulike applikasjonsklassene der P2P-systemer er tatt i bruk. Kategorien "Plattformer" i figur 2.5 svarer ikke til noen spesifikk applikasjonsklasse, og er derfor ikke inkludert i figur 2.6. I den videre diskusjonen gjennomgås de ulike applikasjonsklassene (fra figur 2.6).



Figur 2.6: Klassifikasjon av P2P-applikasjoner [Milojicic et al., 2002].

2.5.1 Parallelliserbare

P2P-teknologi kan utnyttes for å utføre beregninger parallelt. I slike P2P-applikasjoner er prosessorkraft den viktigste ressursen som nodene bidrar med.

Beregningstunge P2P-applikasjoner deler et stort problem opp i mindre og uavhengige delproblemer. Delproblemene blir deretter fordelt mellom samtlige deltakende noder i systemet. Dette medfører at alle nodene utfører de samme beregningene på ulike datasett. Komponentbaserte P2P-applikasjoner skiller seg fra beregningstunge applikasjoner ved at de deltakende nodene utfører forskjellige oppgaver, og dermed må kommunikasjon mellom deltakende noder bli lagt mer vekt på.

Parallelliserbare P2P-applikasjoner i underkategorien "Beregningstunge" er for eksempel knekking av kryptografiske nøkler [distributed.net, 2004] eller signal-analyse av radiobølger [Seti@home, 2004]. Komponentbaserte P2P-applikasjoner er foreløpig ikke blitt realisert, men slike applikasjoner kan baseres på teknologier som JavaBeans eller Web Services [Milojicic et al., 2002].

2.5.2 Fil- og innholdshåndtering

P2P-applikasjoner for fil- og innholdshåndtering utnytter de deltakende nodene sin ledige lagringskapasitet og båndbredde. Applikasjoner for innholdsutveksling lar hver node tilby et utvalg filer til de andre nodene i systemet. Den lokale brukeren på hver node velger hvilke filer han/hun ønsker å tilby andre og hvilke filer han/hun ønsker å laste ned fra andre noder. En variant av innholdsutveksling er P2P-systemer for publisering. Hver bruker kontrollerer selv hvilke dokument han/hun ønsker å publisere i systemet, men når et dokument er blitt publisert vil det automatisk spres utover systemer slik at dokumentet også tilbys fra andre noder enn den opprinnelige.

Filsystemer basert på P2P-teknologi har som formål å innføre konsistent og pålitelig tilgang til ressursene i et P2P-system, men foreløpig befinner slike systemer seg på forskningsstadium. Filtrering- og søkeapplikasjoner har til hensikt å indeksere dataene som er tilgjengelige i P2P-systemet. For å lage et søkbart P2P-nettverk må samtlige noder samarbeide om å bygge en distribuert indeks.

Det finnes et utall applikasjoner for fil- og innholdshåndtering som er tatt i bruk. Gnutella [Gnutella, 2004] og FastTrack [FastTrack, 2004] er eksempler på fildelingsapplikasjoner som er operative. Freenet [Clarke et al., 2001] er en annen P2P-applikasjon som tilbyr anonym publisering av dokumenter. Av filtrerings- og søkeapplikasjoner finnes JXTA Search [Waterhouse, 2001] som er basert på JXTA P2P-plattformen.

2.5.3 Samarbeidsverktøy

Samarbeidsverktøy deles inn i øyeblikksmeldinger (*eng. "instant messages"*), delte applikasjoner og spill. Felles for samtlige underkategorier er at de stiller sanntidskrav til kommunikasjon mellom nodene. Eksempelvis vil det være vanskelig for brukere å føre en dialog ved hjelp av et samarbeidsverktøy med mindre meldingene kommer fram til rett tid og i korrekt rekkefølge.

Øyeblikksmeldinger er blitt et populært applikasjonsområde med svært mange brukere på verdensbasis. Foreløpig er disse tjenestene avhengige av sentrale tjenere, og de benytter seg derfor ikke av P2P-teknologi. Karakteristisk for øyeblikksmeldinger er at en bruker sin tilkoblingsstatus, som for eksempel påkoblet, opptatt eller frakoblet, er synlig for andre brukere i systemet.

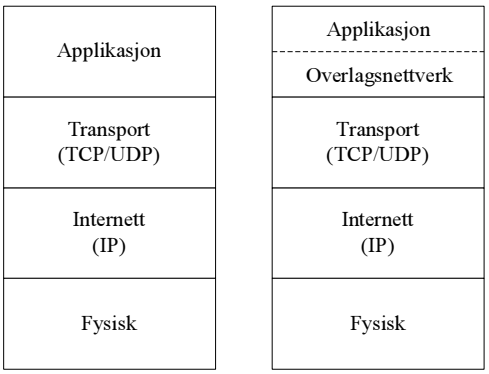
Delte applikasjoner tillater at flere brukere arbeider på samme dokument samtidig, og når en bruker gjør endringer på dokumentet vil samtlige andre brukere se disse endringene øyeblikkelig. Microsoft Netmeeting [Microsoft Corp., 2004] er et eksempel på en applikasjon som har slik funksjonalitet, men baseres likevel på en klient/tjener arkitektur.

Spill med flere deltakere er utbredt på Internett. De fleste spill benytter seg imidlertid av en klient/tjener arkitektur. Net-Z [Quazal Tech., 2004] er en P2P-teknologi som muliggjør onlinespill uten noen form for sentral tjener. Under Net-Z er hver klient ansvarlig for å distribuere oppdateringer av sine egne objekter til de andre spillerne.

2.6 Overlagsnettverk

Et overlagsnettverk (*eng. "overlay network"*) er et virtuelt nettverk som opererer over eksisterende nettverksinfrastruktur [Doval og O'Mahony, 2003]. Tradisjonell nettverkskommunikasjon sees ofte på som en lagdelt tjeneste. Et høy-

erestående lag kan bruke tjenester i lavere lag, mens lavere lag leverer informasjon til høyere lag. Figur 2.7(a) illustrerer en nettverksmodell som ofte kalles TCP/IP-modellen. Det nederste laget i denne modellen er det fysiske nettverkslaget. Dette laget representerer fysiske komponenter i nettverket, blant annet kabler og knutepunkter. Det øverste laget, applikasjonslaget, bruker laverestående tjenester i andre lag for å implementere en bestemt nettverksapplikasjon. Mellom det fysiske og applikasjonslagget er transport- og Internettlaget [Tanenbaum, 1996]. Figur 2.7(b) viser hvordan et overlagsnettverk plasseres i TCP/IP-modellen. Overlagsnettverket er en del av applikasjonslaget, men kan også levere tjenester til andre komponenter. Derfor er overlagsnettverket plassert nederst i applikasjonslaget.



(a) TCP/IP-modellen.

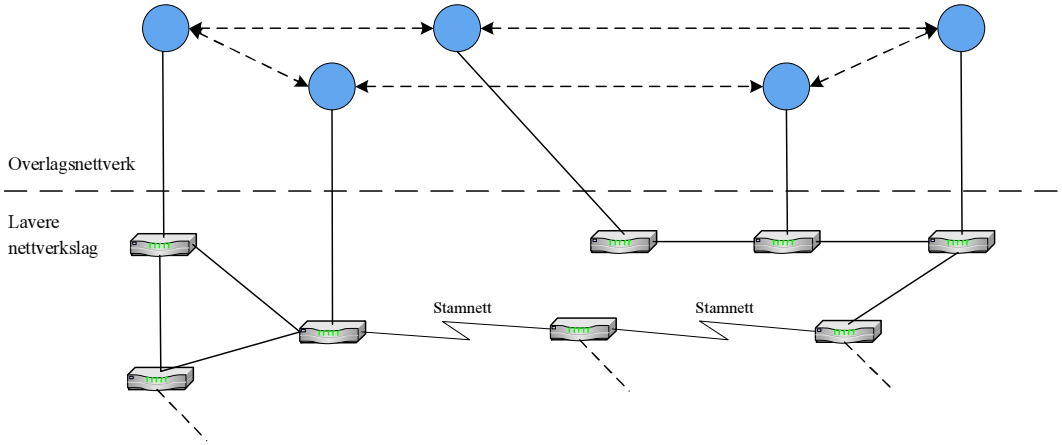
(b) TCP/IP-modell inkludert overlags-nettverk.

Figur 2.7: Plassering av overlaget i forhold til TCP/IP-modellen.

Et overlagsnettverk benytter altså eksisterende nettverksinfrastruktur på Internett for å lage et virtuelt nettverk. Det virtuelle nettverket muliggjør andre funksjoner enn de som allerede er tilgjengelige i de underliggende nettverkslagene. Disse funksjonene kan implementeres uten å måtte modifisere andre nettverkslag, noe som er nødvendig når slike nettverk skal lages over eksisterende infrastruktur på Internett. Å oppgradere eller endre kjernekomponenter i eksisterende infrastruktur er som regel umulig, men med et overlagsnettverk er det likevel mulig å implementere ønsket funksjonalitet.

Noder som deltar i et overlagsnettverk kommuniserer via *virtuelle forbindelser*. Disse forbindelsene kan betraktes som kommunikasjonstunneler som muliggjør kommunikasjon mellom noder i overlagsnettverket. Tunneler er egentlig sti-

er i den underliggende nettverksinfrastrukturen, men overlagsnettverket gjør at det fremstår som en direktesti mellom to noder. En forbindelse karakteriseres ofte ved dens båndbredde, responstid og pakketap. For en tunnel vil disse karakteristikkene være aggregatet av de underliggende nettverksforbindelsene hvor data passerer. En bestemt komponent (for eksempel en ruter) kan delta i flere overlagsnettverk samtidig, og kan ha forskjellige roller i ett overlagsnettverk (både ruter og node). Tunnelene definerer topologien til overlagsnettverket. Overlagsnettverket administreres og kontrolleres av de enkelte nodene i P2P-nettverket. Figur 2.8 viser forholdet mellom overlagsnettverket og den underliggende nettverksinfrastrukturen. Kommunikasjonen i overlagsnettverket skjer "direkte" mellom noder via tunneler, mens kommunikasjonen i lavere lag er indirekte og kan omfatte flere adskilte nettverk.



Figur 2.8: Overlagsnettverk over underliggende infrastruktur.

Overlagsnettverk er en svært fleksibel infrastruktur for å bygge P2P-systemer. Ved å gjøre ruting i applikasjonslaget er det mulig å omgå DNS-systemet, noe som er gunstig for lokasjonsuavhengige nettverkstjenester. Overlagsnettverket er lett å konfigurere og krever ikke spesiell støtte i rutere. Tjenester som multicast, sikkerhet, tjenestekvalitet og adressering kan gjøre bruk av overlagsnettverket i P2P-applikasjoner. Ressurslokalisering og ruting av forespørsler mellom noder er også tjenester som kan implementeres i et overlagsnettverk. Sistnevnte tjenester er svært essensielle for funksjonaliteten til P2P-nettverk, og er videre beskrevet i neste delkapittel.

2.7 Ressurslokalisering og ruting

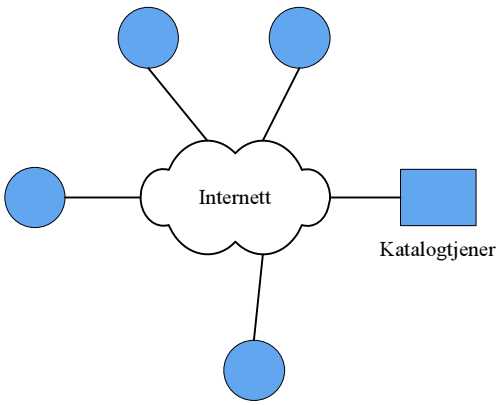
Nettverkstopologien i P2P-systemer er ofte svært dynamisk. Deltakende noder må kunne melde seg inn og ut av nettverket, noe som i praksis betyr at ressurser kommer og går. Ressurslokalisering og ruting av ressursforespørsler er derfor sentrale funksjoner i P2P-nettverk.

En vanlig karakteristikk ved P2P-systemer er at de implementerer en egen ressurslokaliserings- og rutingfunksjon i overlagsnettverket. Siden overlagsnettverket separerer ressurser fra fysisk lokasjon, må ressurser kunne adresseres uavhengig av dens IP-adresse. Hvordan overlagsnettverket organiseres med tanke på topologi og ressurslokaliserings- og rutingmekanismer har stor konsekvens for effektiviteten og skalerbarheten til P2P-systemet. Overlagstopologien bestemmer kommunikasjonskostnaden forbundet med en P2P-applikasjon [Ripeanu, 2001].

[Milojicic et al., 2002] kategoriserer ressurslokaliserings- og rutingtjenestene i overlagsnettverk i tre hovedkategorier: Sentralisert katalogtjeneste, kringkastet forespørsel og dokumentruting. For overlagsnettverk generelt er skalerbarhet en ønsket egenskap. Samtidig er det viktig å kunne gjenfinne alle tilgjengelige ressurser i P2P-nettverket. De forskjellige modellene/kategoriene har vidt forskjellige egenskaper på disse områdene.

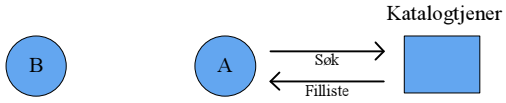
2.7.1 Sentralisert katalogtjeneste

Figur 2.9 illustrerer arkitekturen til en sentralisert katalogtjeneste. I denne modellen brukes en hybrid P2P-arkitektur. Søk etter ressurser gjøres mot en sentral tjener, mens aksess til selve ressursene skjer ved direkte kommunikasjon mellom de deltakende nodene.

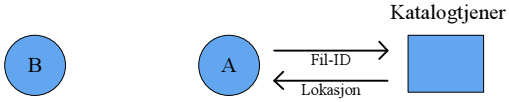


Figur 2.9: Sentralisert katalogtjeneste — arkitektur.

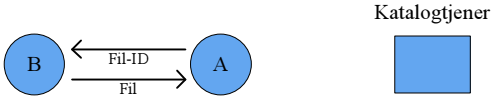
Napster [napster.com, 2004] er et P2P-system som har benyttet en sentralisert katalogtjeneste for å gjøre tilgjengelige ressurser søkbare. Noder som melder seg inn i systemet publiserer sine ressurser til katalogtjenesten. Når en annen node ønsker å finne en ressurs, sender den en forespørsel til katalogtjenesten. Figur 2.10(a) illustrerer denne prosessen. Node A sender et søk til katalogtjeneren som svarer med å returnere en liste over filer som passer søkekriteriet. I neste omgang, illustrert i figur 2.10(b), sender node A en konkret forespørsel for en fil. Tjeneren svarer med å angi adressen (IP-adressen) til noden hvor filen finnes (node B). Deretter tar node A direkte kontakt med node B som har den ønskede filen, og som svarer med å gi node A tilgang (figur 2.10(c)).



(a) Søk



(b) Lokalisering



(c) Nedlasting

Figur 2.10: Sentralisert katalogtjeneste — virkemåte.

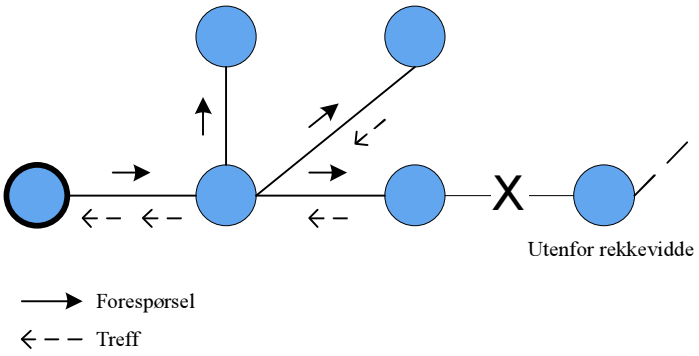
Denne modellen krever en sentralisert infrastruktur som betjener informasjon om deltakerne i systemet. Dette kan medføre begrensninger med hensyn på skalerbarhet fordi flere forespørsler krever raskere tjenere, og større indekser krever at lagringskapasiteten til tjeneren må økes [Milojicic et al., 2002]. Modellen er imidlertid gunstig fordi en deltakernode kun må kjenne til tjenernoden.

2.7.2 Kringkastet forespørsel

Kringkastet forespørsel (*eng. "flooded request"*) har ingen sentraliserte enheter, og er således et ekte P2P-system. Det skjer ingen kunngjøring av delte ressurser

i denne modellen. I stedet kringkastes forespørsler i nettverket. Noder som besitter en ressurs som passer til søkekriteriet sender svar tilbake.

Fordi kringkastede forespørsler generelt vil måtte behandles av samtlige noder i nettverket, er denne metoden svært ressurskrevende. Hver node vil måtte vurdere hver eneste forespørsel i nettverket. For å redusere belastningen på hver deltakernode er det derfor vanlig å bestemme et antall "kringkastingssteg" for en forespørsel. Dette antallet vil for eksempel være i størrelsesorden fem til ni [Milojicic et al., 2002]. Antall ganger en forespørsel videresendes av en node er dermed begrenset. Blant annet Gnutella [Gnutella, 2004] bruker denne metoden. Et eksempel på et søk ved bruk av kringkastet forespørsel er illustrert i figur 2.11. Her initialiserer noden med fet ring en ressursforespørsel som kringkastes i nettverket inntil antall kringkastingssteg (her to) er oppnådd. To av nodene returnerer treff på søket tilbake til initatornoden. En av nodene nås aldri med forespørselen fordi den er utenfor den definerte søkerekkevidden.



Figur 2.11: Kringkastet forespørsel — virkemåte.

Kringkastet forespørsel krever mye båndbredde og skalerer derfor dårlig selv om det ikke finnes sentraliserte enheter. Hvis antall kringkastingssteg begrenses for å redusere båndbreddeforbruket, kan ikke alle ressurser finnes selv om de finnes i nettverket [Milojicic et al., 2002].

2.7.3 Dokumentruting

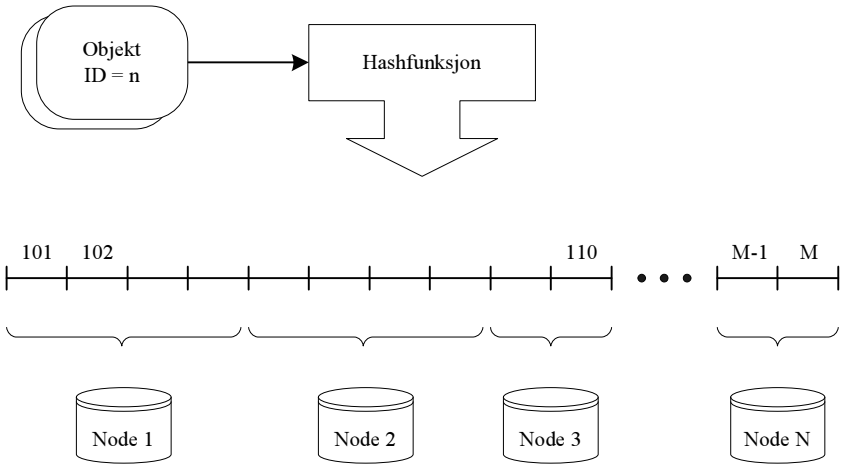
Både sentralisert katalogtjeneste og kringkastet forespørsel har dårlige egenskaper med hensyn på skalerbarhet. En tredje modell, dokumentruting (*eng. "document routing"*), er blant annet implementert i FreeNet [Clarke et al., 2001]. Alle noder og ressurser tildeles unike identifikatorer, og hver node kjenner til et bestemt antall av de andre nodene i nettverket. Identifikatorene brukes for å organisere nodene og ressursene i overlagsnettverket. Strukturen på overlagsnettverket organiseres slik at systemet blir skalerbart samtidig som gjenfinning

av tilgjengelige ressurser garanteres [Milojicic et al., 2002].

Overlagsnettverk som har ressurslokalisering- og rutingmekanismer basert på dokumentruting inkluderer Chord [Stoica et al., 2001], "Content-Addressable Networks" (CAN) [Ratnasamy et al., 2001], Tapestry [Zhao et al., 2001] og Pastry [Rowstron og Druschel, 2001]. To mål er felles for disse systemene:

- Minimere antall hopp/steg som trengs for å rute til ønsket ressurs.
- Minimere mengden informasjon om andre noder i nettverket som hver node må lagre.

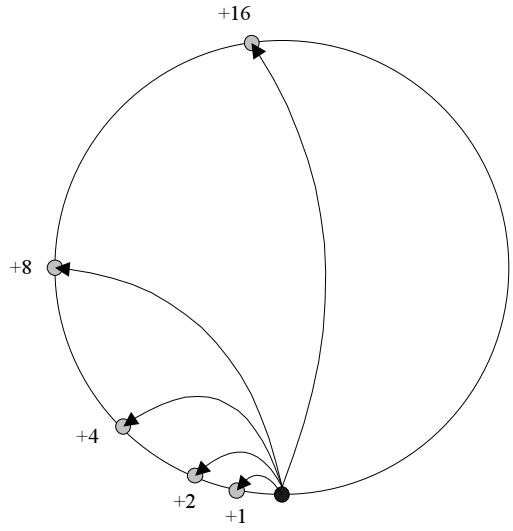
Felles for disse systemene er også at de benytter *distribuerte hashtabeller*. Alle ressurser tildeles en unik ID som er hashverdien av deres innhold og navn. Disse hashverdiene er nøkkelen i en hashtabell som holder orden på ressurser og deres lokasjon. Distribuerte hashtabeller deler ansvaret for å lagre en hashtabell mellom samtlige noder i nettverket. En node har ansvaret for et definert intervall av hashverdier, og andre noder må spørre denne noden for å finne lokasjonen til en bestemt ressurs. Figur 2.12 illustrerer prinsippet med distribuerte hashtabeller.



Figur 2.12: Distribuert hashtabell.

I tillegg til å lagre informasjon om tilgjengelige ressurser i nettverket, må hver node lagre lokasjonsinformasjon om et antall andre noder. Nodeinformasjonen må fordeles slik at det kan garanteres at det finnes en sti av pekere fra en hvilken som helst node til en hvilken som helst annen node. Hvis ikke dette er oppfylt vil ikke alle ressurser kunne lokaliseres av alle noder. I tillegg bør nodene organiseres slik at antall hopp i forbindelse med ruting begrenses.

Chord-modellen organiserer nodene i en ring etter hashverdi. Hver node må lagre lokasjonspekere til $log_2 n$ noder, hvor n er antall noder i nettverket. Figur 2.13 illustrerer et overlagsnettverk organisert etter Chord-modellen og lokasjonspekerne til én av nodene på nettverket.



Figur 2.13: Organisering i Chord-ring.

I Chord-ringen må hver node administrere en rutingtabell. Denne tabellen inneholder lokasjonen (IP-adresse og portnummer) til nodene som ligger halvveis, kvartveis, en åttendedels osv. ganger i omvendt klokkeretning rundt ringen helt til dens etterfølgende nabonode. Altså, det i'te innslaget i rutingtabellen til node x inneholder hashverdien og lokasjonen til noden s gitt ved følgende uttrykk:

$$s = \text{etterfølger}(HV(x) + 2^{i-1}) \tag{2.1}$$

Hvor $1 \leq i \leq m$ (m er antall bit i hashverdien og $HV(x)$ er hashverdien til node x) [Stoica et al., 2001].

I Chord må altså hver node ha $log_2 n$ innslag i rutetabellen, og det tar $O(log_2 n)$ hopp for å lokalisere en ressurs. Når ressursen er lokalisert opprettes det en direkte forbindelse mellom nodene.

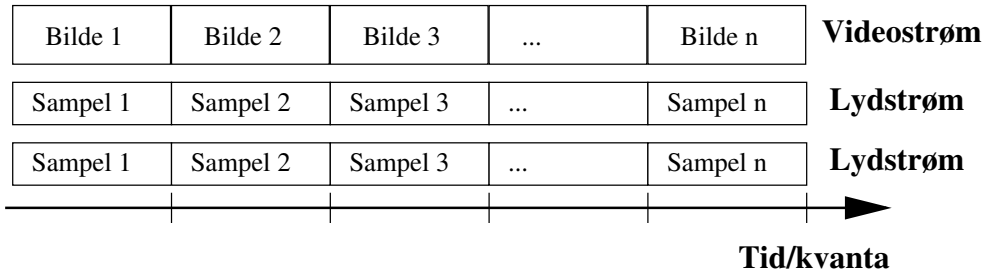
Fordelen med alle de nevnte modellene som er basert på dokumentruting og distribuerte hashtabeller (Chord, CAN, Tapestry, Pastry), er at de er skalerbare og at de kan garantere en maksimalverdi for antall hopp som kreves for å lokalisere en ressurs [Milojicic et al., 2002]. Generelt vil disse modellene være å foretrekke for skalerbare og robuste P2P-systemer selv om både sentralisert

katalogtjeneste og varianter av kringkastet forespørsel har blitt brukt i flere systemer.

Kapittel 3

Kontinuerlige medier

Karakteristisk for digitaliserte kontinuerlige medier er at de har en tidsdimensjon som er delt inn i diskrete tidsintervaller, der hvert intervall er et kvantum av fast eller varierende utbredelse. Video består for eksempel av en sekvens av enkeltbilder, der et bilde tilsvarer den minste enheten langs tidsaksen. En sammensetning av en eller flere kontinuerlige medier omtales som et mediedokument, et medieobjekt eller en mediestrøm. De ulike kontinuerlige media i et mediedokument kalles strømmer, og en vanlig sammensetning av et mediedokument er for eksempel to lydstrømmer og én videostrøm (figur 3.1).

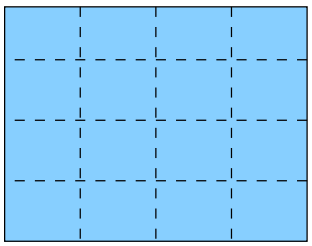


Figur 3.1: Sammensetning av et mediedokument.

3.1 Format

På grunn av at kontinuerlige medier stiller store krav til lagringskapasitet og båndbredde, er det vanlig å benytte teknikker for datakompresjon. "Moving Picture Experts Group" (MPEG) [MPEG, 2004] har publisert flere standardformater for kompresjon av både lyd og video. MPEG-standardene inkluderer i tillegg formater for hvordan flere strømmer kan synkroniseres og multiplekses til én mediestrøm for å kunne transporteres over nettverk [Lu, 1999].

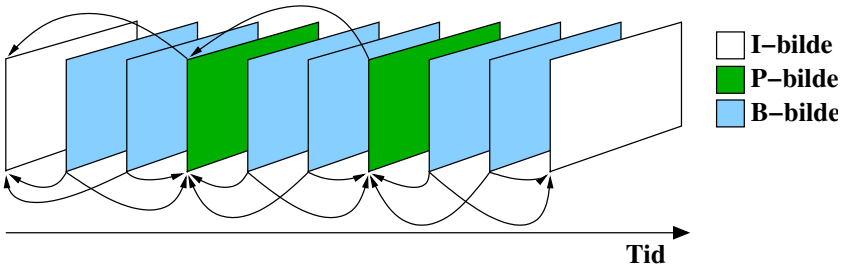
MPEG-standardene for videokompresjon deler hvert enkeltbilde inn i et antall makroblokker (figur 3.2), og et bilde i en MPEG-video består dermed av et sett komprimerte makroblokker. Kompresjonsteknikken som benyttes av MPEG-video kan enten komprimere redundant informasjon internt i et enkeltbilde (romlig kompresjon) eller redundant informasjon mellom ulike enkeltbilder (temporal kompresjon). Romlig kompresjon utføres ved hjelp av en kombinasjon av bildekomprimeringsteknikker og statistiske komprimeringsteknikker. Ved temporal kompresjon lagres kun differansen mellom makroblokker i ulike enkeltbilder, og differansen komprimeres deretter ved hjelp av romlig kompresjon. For hver makroblokk velges en bevegelsesvektor som peker på en makroblokk i et annet enkeltbilde, som differansen beregnes ut fra. En makroblokk kan også ha to bevegelsesvektorer, der den ene peker på en makroblokk i et tidligere bilde og den andre peker på en makroblokk i et framtidig bilde. Differansen til en makroblokk med to bevegelsesvektorer beregnes ut fra gjennomsnittet av makroblokkene som bevegelsesvektorene peker på [Li og Drew, 2004].



Figur 3.2: Inndeling av bilde i makroblokker.

I MPEG grupperes enkeltbildene etter hvordan de er komprimert, og det finnes tre typer enkeltbilder. Figur 3.3 viser hvordan enkeltbildene i en MPEG videostrøm settes sammen. De enkelte bildetypene har følgende egenskaper [Li og Drew, 2004].

- **I-bilder:** Er selvstendige bilder som ikke er avhengige av andre bilder, og de komprimeres derfor kun romlig.
- **P-bilder:** Er avhengige av et tidligere I- eller P-bilde, og temporal kompresjon kan dermed benyttes.
- **B-bilder:** Komprimeres temporalt ut fra tidligere eller framtidige I- eller P-bilder.



Figur 3.3: MPEG-video [Li og Drew, 2004].

B-bilder som er komprimert temporalt mot et framtidig bilde medfører at bildene i en videostrøm ikke forekommer i samme rekkefølge som de spilles av i. I figur 3.3 er de to første B-bildene komprimert mot et framtidig P-bilde, og for å spille av de to første B-bildene må dermed det første P-bildet allerede være overført. Overføringsrekkefølgen medfører krav til bufring under avspilling, og forsinkelsen i avspillingen er derfor avhengig av bufferstørrelsen hos mottaker [Li og Drew, 2004].

3.2 Tjenestekvalitet

Det endelige målet for kvaliteten til en tjeneste for leveranse av kontinuerlige medier er hvordan brukeren opplever avspillingen. Brukeropplevelse er imidlertid vanskelig å måle, og for lettere å kunne vurdere kvaliteten til ulike metoder for leveranse brukes i stedet et sett av målbare parametre:

- **Båndbredde:** Betegner overføringshastigheten som er nødvendig for leveransen. Båndbredde deles inn i gjennomsnittlig og maksimal båndbredde, og måles i kilo- eller megabits per sekund.
- **Forsinkelse:** Overføring av data via nettverk innebærer en forsinkelse fra punkt til punkt, og denne forsinkelsen kan vanligvis måles i millisekunder. Variasjon av forsinkelse (*eng. "jitter"*) er også en svært viktig kvalitetsparameter som gir et mål for hvor jevn leveransen er.
- **Feilrate:** Angir hvor stor andel av trafikken som ikke kommer fram til mottaker. Feilraten avhenger av hvor pålitelig den underliggende nettverksteknologien er, samt hvor mettet kapasiteten er i nettet.
- **Tidsfrister:** For kontinuerlige medier er også andelen av trafikken som ikke ble levert i henhold til tidsfristen interessant.

- **Forskyvelse:** Når en mediestrøm består av flere strømmer, må avspillingen av strømmene være synkronisert innenfor et bestemt tidsintervall. Forskyvelse forekommer ofte når mediestrømmer leveres fra ulike kilder, og dermed følger ruter med ulik forsinkelse gjennom nettverket.

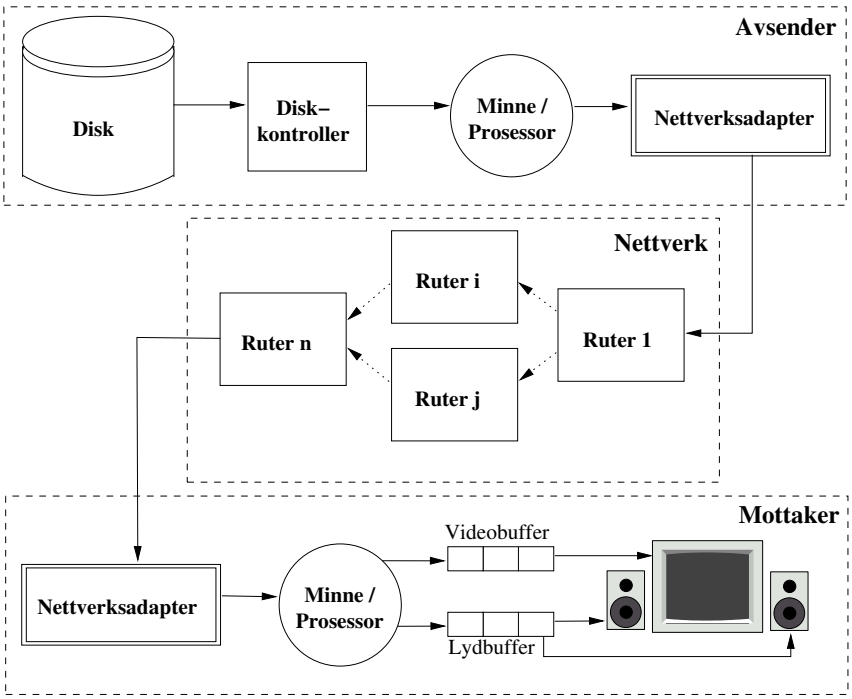
Spesielt for leveranse av kontinuerlige medier er kravene til sanntid og båndbredde [Gemmel et al., 1995]. Sanntidskravet innebærer at hvert kvantum har en tidsfrist som angir når avspillingen av kvantumet må skje innen, og dersom et kvantum ikke er ankommet innen tidsfristen, vil neste kvantum bli spilt av i stedet. Kvanta som ankommer etter at tidsfristen er utløpt blir dermed unyttige, og ordinære pålitelige transportprotokoller som ettersender tapte data er derfor mindre egnet til leveranse av kontinuerlige media.

3.3 Kontinuerlig leveranse

Leveranse av kontinuerlige medier tillater at mottaker starter avspilling mens deler av mediestrømmen fortsatt transporteres i nettverket. Denne prosessen omtales også som streaming, og fordelen med streaming er at mottakeren slipper å vente til hele mediestrømmen er overført før avspillingen kan starte. Avspilling av kontinuerlige medier må foregå i korrekt hastighet, og dersom mediestrømmen består av flere strømmer, må avspillingen av disse synkroniseres slik at for eksempel lyden tilhørende et bilde spilles av samtidig med bildet.

Figur 3.4 viser en forenklet figur over komponentene som er involvert når kontinuerlige medier leveres fra avsender til mottaker. Hver komponent langs datastien har sine egne bufre der innkommende data blir lagret midlertidig før de blir sendt videre til neste komponent. De siste bufferne i datastien er mottakeren sine lyd- og videobufre. Mottaker spiller av data som ligger i disse bufferne, og avspillingen av henholdsvis lyd og video vil derfor opphøre dersom disse bufferne går tomme. Kontinuerlig leveranse oppnås dersom ingen av bufferne i komponentene langs datastien tømmes, det vil si at ingen av komponentene sultefôres. Dette tilsvarer at forsinkelsen for videresendelse i en komponent er mindre eller lik forsinkelsen i neste komponent. For å oppnå kontinuerlig leveranse brukes følgende teknikker [Sitaram og Dan, 2000].

- Bufring.
- Planlegging.
- Reservasjon av ressurser.



Figur 3.4: Datasti ved leveranse.

Bufring er en teknikk som brukes for å oppnå en jevn avspillingsrate, og er derfor svært viktig for mottaker. Nettverkskommunikasjon kan foregå med ujevn hastighet der data overføres med høy hastighet i korte etterfølgende perioder. For å jevne ut variansen i hastigheten oppretter mottaker en buffer i minneområdet som den innkommende mediestrømmen lagres i. Avspillingen foregår med jevn hastighet, og programvaren som spiller av mediestrømmen henter alle data fra bufferen. Så lenge bufferen verken er tom eller full, kan selve leveransen utføres med ujevn hastighet samtidig som mediestrømmen spilles av med jevn hastighet. Dersom bufferen tømmes for data, vil avspillingen opphøre inntil data igjen kan hentes ut fra bufferen. I det motsatte tilfellet, der bufferen er full, er ikke mottaker i stand til å lagre innkommende data i bufferen, og leveransen opphører dermed midlertidig uten at det nødvendigvis oppstår avbrudd i selve avspillingen.

Ressursreservasjon innebærer reservasjon av båndbredde, minnekapasitet og prosessorkraft i hver komponent langs datastien. I tillegg kan ressursreservering inkludere oppsett av selve datastien med reservasjon av ressurser i hver nettverksruter som trafikken skal innom. Ved å reservere ressurser sikres tilstrekkelig kapasitet for leveransen. Dagens Internett har en stor mangel i form av at det ikke støtter reservasjon av ressurser, og konsekvensene av dette er at ingen leveransegarantier kan gis. Reservasjon av ressurser i avsenderens komponenter kan utføres, men krever gjerne operativsystemer spesielt tilpasset

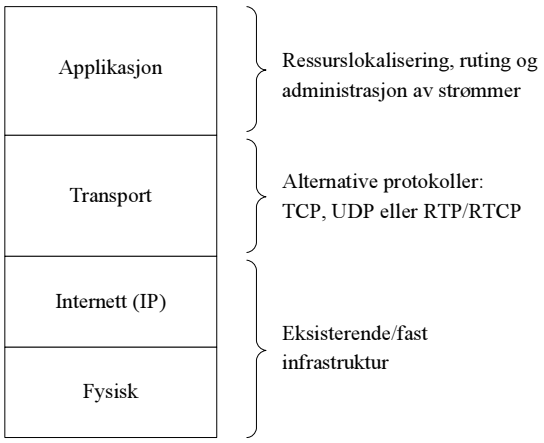
multimedia.

Planlegging brukes for å velge hvilken pakke av strømmen som skal overføres videre til neste komponent. Hver pakke har en tidsfrist for avspilling, og planleggingen må utføres slik at alle pakker videresendes i henhold til tidsfristene. I lagringssystemet til avsenderen utføres planlegging før blokker hentes ut fra disk, og planlegging skjer også før pakker sendes til nettverksadapteret. Videre må hver ruter i nettverket også planlegge hvilke pakker som til ethvert tidspunkt skal videresendes.

3.4 Nettverkskommunikasjon

Tjenestekvalitet er et vidtspennende begrep som omfatter alle parametre som påvirker utførselen av en tjeneste, og felles for disse parametrene er at de beskriver egenskaper som tidsgarantier, kapasitet eller tilgjengelighet. Parametre som spesifiserer tjenestekvalitet finnes i alle komponenter og samtlige lag som en tjeneste består eller benytter seg av. Den oppnåelige tjenestekvaliteten i ovenforliggende lag i nettverksmodellen avgjøres i stor grad av hva som er oppnåelig i de underliggende lagene. Tjenestekvaliteten til en videotjener vil for eksempel være begrenset av tjenestekvaliteten til delsystemer som nettverkskommunikasjon, lagringssystem og operativsystem.

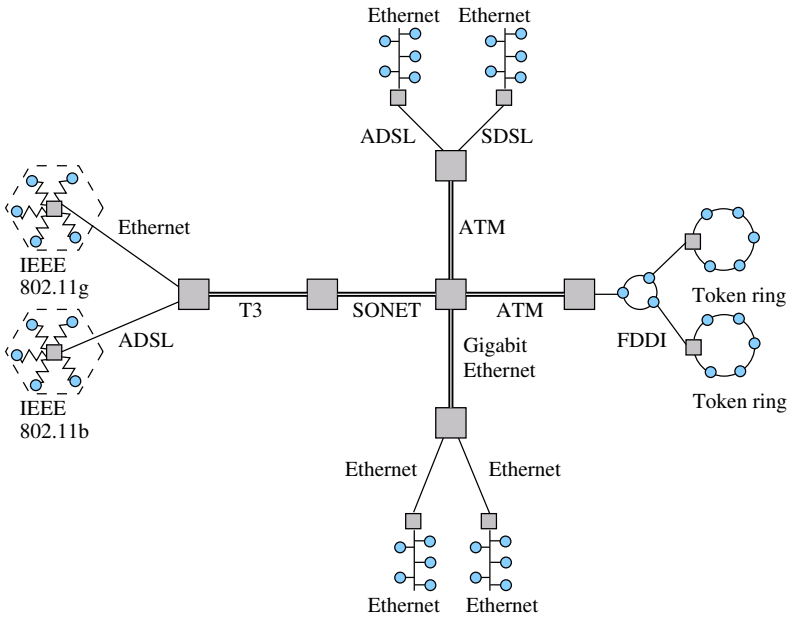
Tjenestekvalitet i nettverk for datakommunikasjon baseres på parametrene som støttes av de lavereliggende lagene i protokollstakken (figur 3.5). Dersom det fysiske laget har støtte for tjenestekvalitet, må Internett-laget tilby å konfigurere denne tjenesten, slik at overliggende lag som transport og applikasjon kan benytte seg av garantiene som gis av det fysiske laget. Dette er imidlertid sjelden tilfelle i dag, der Internett-laget mangler slik støtte, og det lille som finnes av tjenestekvalitet tilbys derfor av applikasjonslaget.



Figur 3.5: Lag av nettverksprotokoller.

3.4.1 Nettverkslaget (Internett)

Den opprinnelige målsetningen for utviklingen av Internett-teknologien var å realisere nettverksprotokoller som kunne knytte sammen eksisterende nettverk. For at dette skulle fungere måtte "Internet Protocol" (IP) være uavhengig av den fysiske nettverksteknologien [Clark, 1988]. Dette har medført at IP har blitt tatt i bruk uten at det har vært nødvendig å skifte ut eksisterende nettverksinfrastruktur. En direkte konsekvens av dette er at tjenestekvalitet i IP fortsatt ikke er innført, og dette på tross av at flere løsninger for tjenestekvalitet i IP eksisterer [Zhao et al., 2000]. Problemet bunner i det faktum at tjenestekvalitet i et logisk nettverkslag støtter seg på de former for tjenestekvalitet som gis av nettverksteknologien som pakkene faktisk transporteres over. Enkelte nettverksteknologier mangler støtte for tjenestekvalitet, og siden Internett-protokollene må kunne fungere med samtlige typer underliggende nettverkslag, baseres derfor IP på en generell antagelse om fullstendig mangel på tjenestekvalitet i lavereliggende lag.



Figur 3.6: Underliggende nettverksteknologi.

Et annet problem er å gi garantier om tjenestekvalitet fra ende til ende. Ofte transporteres et IP-datagram over flere forskjellige nettverksteknologier, og en forutsetning for å kunne gi tjenestegarantier fra ende til ende er at samtlige underliggende nettverksteknologier støtter et sett med kompatible parametre for tjenestekvalitet. Figur 3.6 illustrerer hvilke underliggende nettverksteknologier et utsnitt av Internett *kan* benytte seg av. I figuren er Ethernet og ATM eksempler på nettverksteknologier som benyttes i Internett med henholdsvis ingen og omfattende støtte for tjenestekvalitet.

Nettverkslaget sine ytelsesparametre bestemmes i all hovedsak av det fysiske laget sine egenskaper, og tabell 3.1 gir en oversikt over de viktigste av disse.

Egenskap	**Beskrivelse**
Båndbredde	Måles i (kilo, mega, giga) bits per sekund.
Forbindelses-orientering	Forbindelsesorientering innebærer at en forbindelse mellom endepunktene må opprettes før overføringen kan starte. Et forbindelsesløst nettverk trenger ingen forbindelse, og sender i stedet enkeltstående pakker mellom endesystemene.
Forsinkelse	Tiden som forløper når data sendes fra ende til ende.
Jitter	Angir variasjon i forsinkelsen.
Kringkasting	Multicast-trafikk vil ikke oppta mer båndbredde enn unicast-trafikk dersom nettverksteknologien har direkte støtte for kringkasting.
Pakketap/bitfeil	Sannsynligheten for at en pakke eller bit blir feilaktig overført eller ikke blir overført i det hele tatt.
Pakke/ linjesvitsjing	I et pakkesvitsjet nett kan pakker følge ulike ruter gjennom nettet, og pakker vil derfor kunne ha ulik forsinkelse og ankomme mottaker i feil rekkefølge. I et linjesvitsjet nettverk følger alle data samme rute gjennom nettverket, og de ankommer derfor i korrekt rekkefølge og med tilnærmet lik forsinkelse.
Pålitelighet	Angir om nettverksteknologien garanterer at data som skal sendes vil ankomme mottaker.
Tjenestekvalitet	Hovedkategoriene er ingen (*eng. "best-effort"*), prioritet per pakke og garantier per forbindelse.

Tabell 3.1: Parametre i underliggende lag som påvirker nettverkslaget.

3.4.2 Transportlaget

Transportlaget brukes for å kontrollere og multiplekse overføring mellom to endepunkter. Blant Internett-protokollene finnes to generelle transportprotokoller, "User Datagram Protocol" (UDP) og "Transport Control Protocol" (TCP). UDP er en meget enkel forbindelsesløs transportprotokoll som sørger for at overføringsfeil kan oppdages og at overførte pakker kan settes sammen i den opprinnelige rekkefølgen. TCP er derimot kompleks, og tilbyr forbindel-

sesorientert transport mellom to endepunkter. Viktige oppgaver som utføres av TCP inkluderer:

- **Korrekthet:** Ved hjelp av kontrollsummer sjekker transportlaget at de dataene som ankommer mottaker er de samme som ble sendt fra avsender.
- **Pålitelighet:** Alle pakker som ankommer mottaker må bekreftes mottatt. Dersom en pakke går tapt underveis, og dermed ikke bekreftes mottatt, må transportlaget sørge for at den tapte pakken blir sendt på ny.
- **Rekkefølge:** Pakker kan ankomme i en annen rekkefølge enn den de ble sendt ut i. Transportlaget sørger for at pakkene blir satt sammen i korrekt rekkefølge.
- **Metningskontroll:** Ved overskridelse av kapasiteten i det fysiske laget vil pakker gå tapt. Transportlaget utøver metningskontroll for at det fysiske nettverkslaget ikke skal bli overbelastet, og dermed holdes pakketapet på et minimumsnivå.
- **Multipleksing:** For å kunne skille pakker som hører til forskjellige tilkoblinger mellom to endepunkter utfører transportlaget multipleksing av trafikken.
- **Flytkontroll:** Avsender og mottaker kan ha ulik kapasitet, og for at avsender ikke skal sende pakker med høyere rate enn hva mottaker har anledning til å håndtere, må transportlaget utøve flytkontroll.

Leveranse av kontinuerlige medier stiller spesielle krav til transportprotokoller. Kravene inkluderer støtte for pakketransport i sanntid, transportkontroll, transportovervåkning og tilkoblingsoppsett. Sanntidsleveranse innebærer at pakker må komme fram i rett tidsintervall. Dersom pakker leveres for raskt er det ikke sikkert at klienten har kapasitet til å bufre dem før de skal spilles av. I det motsatte tilfellet, der pakker leveres for tregt, vil ikke mottaker ha data å spille av, og sluttbrukeren opplever et avbrudd i avspillingen.

Pakketap i forbindelse med sanntidskommunikasjon må unngås i stedet for å korrigeres i ettertid. Dersom et videobilde ikke kommer fram til rett tid er det ingen vits i å ettersende det senere. Mottaker må i slike tilfeller hoppe over det manglende bildet og heller fortsette med å spille av resten av videoen. De spesielle kravene medfører at kontinuerlig leveranse utføres av to protokoller, RTP og RTCP, der RTP utfører selve leveransen mens RTCP kontrollerer at leveransen går riktig for seg.

Real Time Protocol "Real Time Protocol" (RTP) er transportprotokollen for kontinuerlige medier blant "Internet Engineering Task Force" (IETF) sine protokoller, og er definert i [Schulzrinne et al., 2003]. Det er verdt å merke seg er at RTP ikke er én fast definert protokoll. RTP består snarere av et rammeverk med funksjonalitet som kan benyttes for å lage sanntidsprotokoller. For å bruke RTP til for eksempel streaming av lyd og video i MPEG-2 format, må protokollen utformes etter RTP-profilen for lyd og video. Videre spesifiseres innsetting av MPEG-2 data i RTP-pakker, samt spesielle feltverdier i RTP pakkene som er spesifikke for MPEG-2, i et eget nyttelastdokument (*eng. "payload document"*). RTP kan altså tilpasses i to nivåer, og det ene nivået (profil) angir hvordan RTP må tilpasses en spesiell tjeneste. Det neste nivået (nyttelast) angir hvordan en RTP-profil må tilpasses et spesifikt format for dataene som tjenesten vil benytte seg av.

RTP benytter seg av lavere lag for transport av data, og i praksis medfører sanntidskravene at RTP benytter seg av UDP sine kontrollsum- og multipleksingstjenester. RTP er plassert mellom lagene applikasjon og transport i protokollhierarkiet. På grunn av at RTP må tilpasses hver tjeneste, blir protokollen ofte implementert som en del av en applikasjon, i motsetning til lavereliggende lag som blir tilbudt som en tjeneste fra operativsystemet.

RTP tilbyr følgende tjenester til applikasjonen:

- Typeidentifisering — hva pakkene inneholder.
- Sekvensnummerering — økes for hver pakke.
- Tidsstempling — økes for hvert kvantum.

Typeidentifisering brukes for å identifisere type og format av innholdet i pakkene, og dette spesifiseres i profildokumentet til den aktuelle tjenesten. Hver RTP-pakke bærer et sekvensnummer som gjør at RTP kan gjenopprette den opprinnelige rekkefølgen av innkomne pakker før de overleveres til applikasjonen. Dette sekvensnummeret kan også brukes til deteksjon av pakketap.

Hver RTP-pakke inneholder også et felt som angir når den skal spilles av, det vil si tidspunktet for når avspilling av de første dataene i pakken skal starte. I tilfeller der for eksempel lyd og video sendes som separate mediestrømmer, vil tidsstemplene også bli brukt for å synkronisere disse. Selv om RTP er en protokoll for transport av sanntidsdata, kan den ikke gi noen flere tjenestegarantier enn det som gis av det lavereliggende nettverkslaget. RTP bruker en egen kontrollprotokoll (RTCP) for å sikre så god leveranse som mulig.

Real Time Control Protocol Mellom sender og mottaker av en RTP-datastrøm sendes kontrollinformasjon i begge retninger, og til dette brukes

"Real Time Control Protocol" (RTCP). Hovedhensikten med RTCP er å sende tilbakemeldinger om sending og mottak, og på bakgrunn av tilbakemeldingene kontrollere flyt og metning i nettet. Tilbakemeldingen brukes også til å beregne pakketap, gjennomsnittlig forsinkelse og variasjon i forsinkelsen . Inkludert i RTCP-pakker er også et felt som identifiserer avsender, og vanligvis angir dette bruker- og maskinnavn. Ellers brukes RTCP også til å angi koblingen mellom logisk kvantumtid og tidspunktet for når et kvantum faktisk skal spilles av.

En viktig forutsetning for at RTCP skal fungere er at den ikke opptar for mye av trafikken, og maksimalgrensen er satt til fem prosent av den totale båndbredden som brukes til transport av selve mediestrømmen. Mesteparten av kompleksiteten i RTCP er forbundet med skalering av multicast-videokonferanser. Når antallet deltakere i en konferanse øker vil for eksempel de eksisterende deltakerne midlertidig holde tilbake sine RTCP-pakker, og dermed unngås en kortsiktig men kraftig eskalering av RTCP-trafikken. Disse reglene samt hvilke algoritmer som skal brukes er spesifisert i [Schulzrinne et al., 2003].

Kapittel 4

Streaming i P2P-nettverk

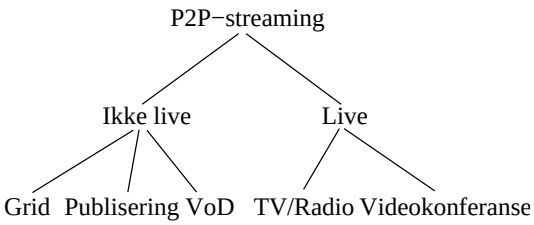
Tidligere har systemer for leveranse av kontinuerlige medier som lyd og video vært basert på klient/tjener-modellen. Dette har medført at distribusjon av slike medier har stilt store krav til leverandørens infrastruktur samtidig som løsningene ikke har vært særlig skalerbare.

P2P-systemer for streaming kan anvendes både som et alternativ til og i tillegg til eksisterende teknologi. Enkelte tjenester som tidligere har vært utført av nettverkslaget, kan ved hjelp av P2P-teknologi utføres i applikasjonslaget. Variasjon i applikasjonenes formål eller operative omgivelser kan medføre at de funksjonelle egenskapene ved P2P vektes forskjellig.

Det er ønskelig at P2P-baserte streamingtjenester skal ha tilsvarende kvalitet som klient/tjener-baserte tjenester tilbyr under normale forhold. På grunn av at P2P-systemer ofte opererer i upålitelige og lite forutsigbare omgivelser, blir dette imidlertid vanskelig. For å kunne forbedre tjenestekvaliteten til streaming i P2P-nettverk er det viktig å kartlegge hvilke faktorer som den påvirkes av.

4.1 Applikasjonsområder

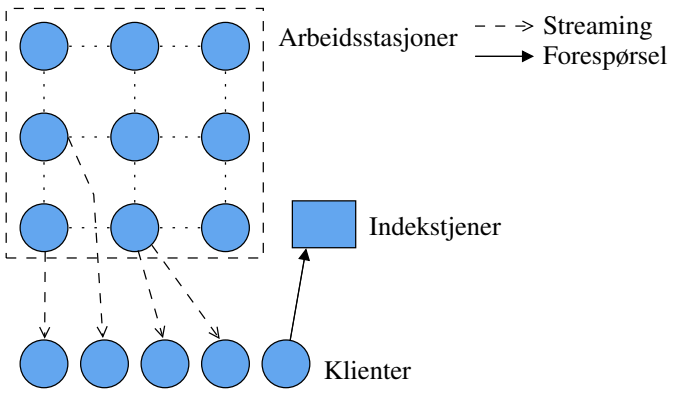
P2P-teknologi og streaming kan anvendes innen flere ulike applikasjonsområder. Figur 4.1 viser viktige applikasjonsområder, og i figuren skilles applikasjonsområdene alt etter om mediet streames live eller ikke.



Figur 4.1: P2P-applikasjoner med kontinuerlige medier.

4.1.1 Virtuell videotjener

Større organisasjoner kan utnytte kapasiteten i arbeidsstasjoner som ikke er i bruk til å distribuere kontinuerlige medier. Et universitet kan for eksempel utnytte ledige ressurser til å distribuere video fra undervisning. Figur 4.2 viser et eksempel på hvordan en samling arbeidsstasjoner kan opptre som en stor virtuell videotjener. For å kunne holde orden på ressursene som er tilgjengelige kan en indekstjener brukes, og løsningen vil dermed ha likhetstrekk med hybride P2P-systemer. Når en klient ønsker å koble seg til den virtuelle tjeneren sender klienten først en forespørsel til indekstjeneren. Indekstjeneren har full oversikt over ledige ressurser og oppgaver som utføres av hver node, og indekstjeneren kan derfor velge den noden som er best egnet til å levere et gitt medieobjekt. Et slikt system må kunne replikere medieobjekter jevnt utover nodene, og replikeringsløsningen bør også ta hensyn til lastbalansering ved å replikere populære objekter slik at de til enhver tid er tilgjengelige for klienter. Siden indekstjeneren har oversikt over ressursene i systemet kan den også utføre replikeringen, men det er også mulig å tenke seg en mer desentralisert løsning der hver node selv er ansvarlig for å replikere sine egne medieobjekter.



Figur 4.2: Virtuell videotjener.

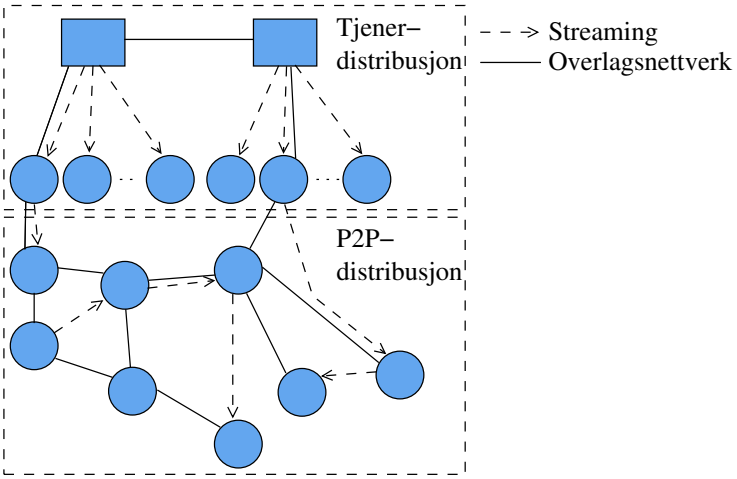
Dersom alle arbeidsstasjonene administreres av samme organisasjon vil et slikt system ha forutsigbare og kontrollerbare omgivelser, og det vil for eksempel være mulig å overvåke ressursbruken til systemet. En felles administrasjon medfører også at nodene kan bli pålagt til å delta i systemet. I tillegg har eierorganisasjonen muligheten til å påvirke omgivelsene til systemet, og det er derfor mulig å stille strenge krav til ytelsesparametre som båndbredde, prosessorkraft og lagringskapasitet til de deltakende nodene.

4.1.2 Video på forespørsel

Tradisjonelle løsninger for å levere video på forespørsel (*eng. "Video on Demand", VoD*) baserer seg på klient/tjener-modellen. I et slikt system leverer et antall tjenere video til klientene, og hver klient forbruker dermed ressurser i en tjener. En videotjener må være i stand til å betjene et stort antall klienter, og stiller derfor strenge ytelseskrav til både maskin- og programvare. Problemet med klient/tjener-modellen er at kapasiteten begrenses av ytelsen til tjenerne, og dermed ikke øker i takt med antall klienter. Kostbare videotjenere medfører at det i praksis er vanskelig å få klient/tjener-baserte systemer for videoleveranse til å skalere.

Siden tjenester på Internett har et globalt publikum, kan de også potensielt tiltrekke seg et større antall klienter enn hva som er mulig å betjene, og tjenesten forblir dermed utilgjengelig på grunn av overbelastning. Dette fenomenet omtales som "flash crowds" eller "slashdot-effekten". Backslash er et P2P-basert system der ordinære web-tjenere, som er i fare for å overbelastes, kan avlastes ved å benytte seg av et P2P-nettverk for å distribuere innhold [Stading et al., 2002].

En mulig løsning på skalerbarhetsproblemet til videotjenere er å innføre et lignende system med en P2P-basert distribusjonsmodell for video. Dersom klientene er utstyrt med P2P-programvare, kan videotjenerne utnytte dette til å iverksette P2P-distribusjon. Figur 4.3 viser en situasjon der videotjenerne betjener et maksimalt antall klienter, og for at systemet skal kunne fortsette å akseptere forespørsler fra nye klienter, er P2P-systemet også aktivert.



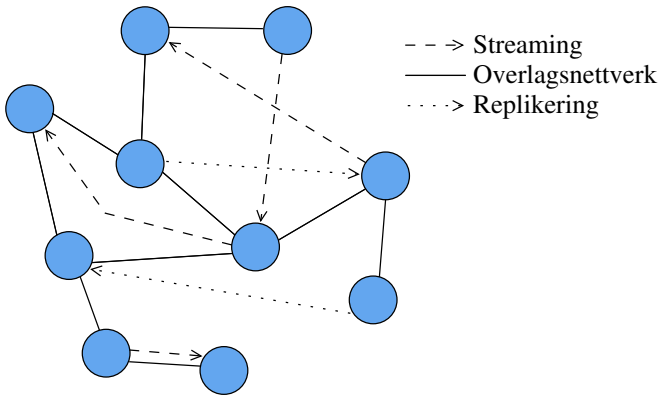
Figur 4.3: P2P kombinert med tjenerdistribusjon.

Dette innebærer at klienter som tidligere har spilt av et medieobjekt, og dermed også lagret det, forblir tilgjengelige slik at videotjenerene kan iverksette P2P-distribusjon med disse tidligere klientene som kilder. I figuren er også videotjenerne med i overlagsnettverket med den hensikt at de skal fungere som et kjent aksesspunkt som nye noder kan benytte seg av når de melder seg inn.

4.1.3 Publisering

Eksisterende P2P-systemer for fildeling kan utvides til også å støtte streaming av kontinuerlige medier. Et slikt system tillater dermed publisering av mediedokumenter uten strenge krav til infrastruktur. Dermed får også privatpersoner og frivillige organisasjoner med begrensede midler tilgang til å publisere mediedokumenter uten å måtte investere i høyytelses-infrastruktur som videotjenere og kommunikasjonsutstyr. Kostnadene fordeles i stedet på de deltakende nodene i P2P-nettverket. Det er tenkelig at et slikt system har en form for automatisk replikering slik at hvert medieobjekt vil eksistere i flere replikater på forskjellige noder. I tillegg bør noder som har spilt av et medieobjekt lagre det, for senere å kunne tilby det samme objektet til andre noder.

Figur 4.4 viser et eksempel fra et P2P-system for publisering av kontinuerlige medier. For at samtlige deltakere skal opptre som likeverdige er det hensiktsmessig å unngå enhver form for sentral kontroll, og derfor bør et slikt system benytte seg av et desentralisert overlagsnettverk.



Figur 4.4: Publisering av mediedokumenter.

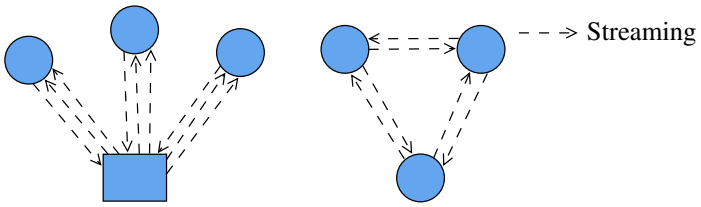
4.1.4 TV/radio-stasjon

Multicast-systemer kan baseres på P2P-teknologi for å realiseres i applikasjonslaget (beskrevet i 4.2). Kravet for å anvende slik teknologi er at kilden har tilstrekkelig båndbredde til å levere mediestrømmen til minst én mottaker, og motivasjonen for å anvende multicast er at teknologien stiller lave krav til infrastruktur hos kilden. En slik tjeneste egner seg til livedistribusjon der en mediestrøm distribueres til mange mottakere. Multicast gir dermed alle med ordinær tilgang til Internett muligheten til å opprette tjenester som for eksempel TV/radio-stasjoner.

4.1.5 Videokonferanse

De fleste systemer for videokonferanser baseres på klient/tjener-modellen (figur 4.5(a)) der hver deltaker sender sin video- og lydstrøm til en sentral tjener, og hvor tjeneren igjen videredistribuerer hver deltaker sin mediestrøm til de andre deltakerne. Ulempen med en slik løsning er at mesteparten av systemets forbrukte båndbredde belastes tjeneren.

I et system for videokonferanse basert på P2P (figur 4.5(b)) unngås en sentral tjener, men hver deltaker må ha tilstrekkelig båndbredde til å sende *én* kopi av sin mediestrøm til *hver* av de andre deltakerne. Fordelen med P2P-baserte videokonferanser er at de ikke avhenger av eksisterende tjenere og derfor enklere kan opprettes spontant mellom en gruppe deltakere.



(a) Klient/tjener (b) P2P

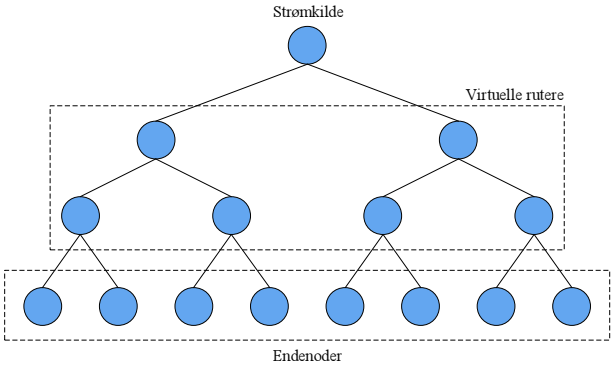
Figur 4.5: Alternative modeller for videokonferanse.

4.2 Multicast

Multicast betyr å adressere og sende data til flere dedikerte noder i et nettverk der avsenderen sender dataene én gang, og hvor ruterne i nettverket tar seg av å sende dem til de riktige mottakerne. Kun de adresserte mottakerne mottar data, i motsetning til ved kringkasting hvor samtlige deltakere i nettverket mottar dataene. Multicast gjør at en node på nettverket kan sende data til flere enn én node uten at det stiller høyere krav til avsenders båndbredde. IP har i utgangspunktet støtte for multicast ved bruk av IP-adresser tilhørende klasse D (hver adresse identifiserer en gruppe noder) [Tanenbaum, 1996]. Det er imidlertid få nettverk som støtter IP-multicast, blant annet fordi det krever tilstander i nettverket ved at hver ruter må ha en rutertabell som forteller hvilke noder som deltar i bestemte multicast-grupper. Båndbreddemessig er imidlertid datadistribusjon mer skalerbart med multicast i forhold til unicast (én til én kommunikasjon).

Å være i stand til å levere en mediestrøm til mange brukere vil være nyttig for en rekke applikasjoner. Nettradio og TV over Internett er begge direktesendte tjenester som krever høy båndbredde. Med IP-multicast ville båndbreddekravet, og dermed driftskostnaden for en innholdsleverandør, blitt betydelig lavere.

I stedet for å ha en fullt sentralisert tjeneste og betale for mye båndbredde, er det mulig å organisere nettverket slik at alle mottakere av en strøm også bidrar med ressurser ved å videredistribuere strømmen. Figur 4.6 viser hvordan et P2P-system kan benyttes for å utføre multicast-ruting i applikasjonslaget.



Figur 4.6: Multicast i overlagsnettverket.

Samtlige noder, med unntak av strømkilden, er konsumenter i multicast-treet. Hver enkelt node kan bli ansvarlig for å videresende strømmen til et antall andre noder i treet. I figuren er det noen noder som både er konsumenter og multicast-rutere. Andre noder er bare konsumenter, men disse kan bli ansvarlige for å videresende strømmen når nye noder melder seg inn i systemet.

Fordelen med multicast i et P2P-system er at det muliggjør utsendelse av mediestrømmer til flere noder enn om strømkilden måtte sende én dedikert strøm til hver node. Det er imidlertid flere utfordringer og problemer knyttet til denne løsningen. I et P2P-nettverk, hvor noder når som helst kan melde seg ut av nettverket eller bli utilgjengelige av andre årsaker, vil ruternoder være kritiske enheter som potensielt kan ødelegge leveransen for dets etterkommere i treet. Et annet problem med multicast som følger en tre-topologi, er at det er mange noder som ikke bidrar med ressurser i nettverket. Gitt et balansert tre med forgreningsfaktor f og høyde h. I et slikt tre vil det være $\frac{f^h-1}{f-1}$ "interne" noder (rutere) og f^h løvnoder i treet. I et binærtre vil over halvparten av nodene i treet være løvnoder, det vil si noder som ikke bidrar med ressurser. Jo høyere forgreningsfaktor i treet, desto mindre andel av nodene bidrar med ressurser [Castro et al., 2003]. Et mer ustrukturert kommunikasjonsmønster, for eksempel en graf-topologi, vil kunne nyttegjøre en større andel av de tilgjengelige ressursene i nettverket. Et eksempel på et slikt system er "Splitstream" [Castro et al., 2003].

4.3 Adaptivitet

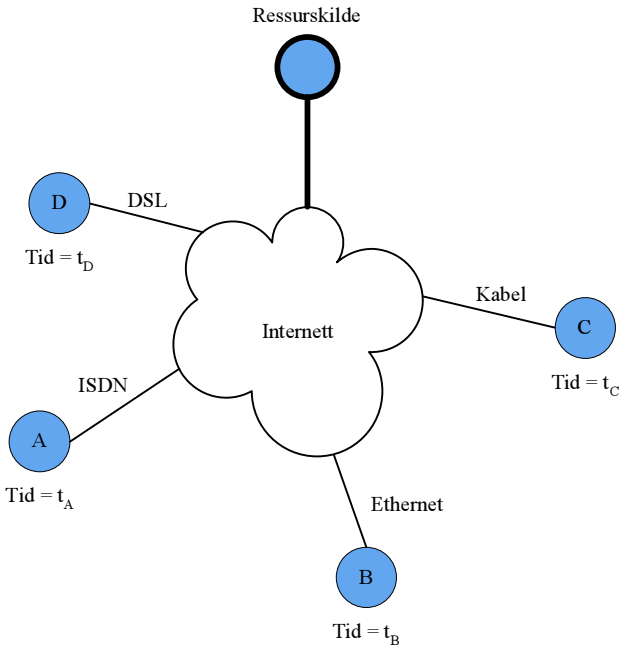
Å streame sanntidsdata (som lyd og video) i et P2P-nettverk er en utfordring av flere årsaker:

- **Ulik nettverksbåndbredde:** Nodene i et P2P-system er datamaskiner

på randen av Internett, og befinner seg derfor ofte i kommersielle operatørnettverk. Båndbredden som den enkelte nettverksoperatør tilbyr kan variere, noe som gjør at forskjellige noder har ulik nettverksytelse.

- **Tilfeldige forespørsler:** I et P2P-nettverk er hver node en uavhengig enhet. En node kan forespørre en ressurs når som helst, og det er derfor vanskelig å synkronisere leveranse av ressurser. En enkel multicast-strøm kan derfor ikke alltid møte kravene til alle noder i systemet.
- **Dynamisk nettverkstopologi:** Fordi enhver node kan forlate nettverket, kan ressurskilder forsvinne selv om den leverer data til andre noder. Ressurser kan derfor forsvinne mens de er i bruk.

Figur 4.7 illustrerer asynkrone forespørsler i omgivelser med heterogen nettverksbåndbredde. Nodene A, B, C og D ønsker alle å aksessere samme ressurs, men til ulike tider (henholdsvis t_A, t_B, t_C og t_D), og befinner seg på forskjellige nettverkstyper. Båndbreddemessig har eksempelvis ISDN (64 Kb/s) langt dårligere båndbredde enn Ethernet (10-10000 Mb/s).



Figur 4.7: Heterogene nettverksomgivelser og tilfeldige forespørsler.

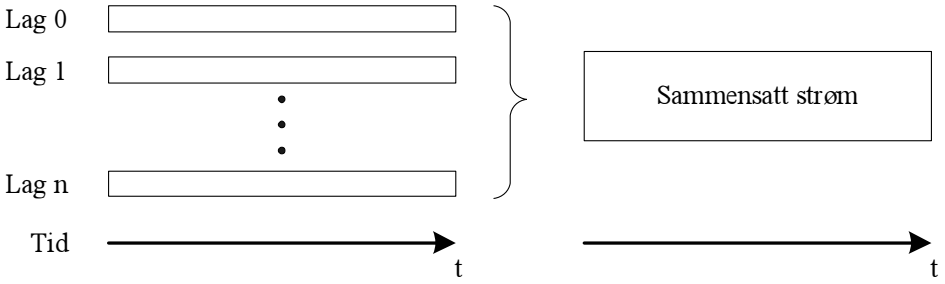
Problemet i slike omgivelser er at ressurskilden i utgangspunktet må sende individuelle strømmer til hver node — selv om noen forespørsler overlapper hverandre i tid. I tillegg har hver mottaker ulike egenskaper med tanke på hva

den er i stand til å motta. Noden på ISDN-nettverket er ikke i stand til å motta en strøm med båndbredde høyere enn 64 Kb/s, mens noden på Ethernet teoretisk kan motta en strøm med tusen ganger høyere båndbredde. For ressurskilden betyr dette at hver node har ulike forventninger/krav til kvaliteten på strømmen den skal motta. Det er derfor ønskelig at ressurskilden kan levere en adaptiv tjeneste, det vil si at den er i stand til å tilpasse båndbredden på en strøm etter hvilken node den leverer den til. I perioder med høy last for en ressurskilde (mange overlappende forespørsler), kan adaptivitet utnyttes for å tilby en tjeneste med litt lavere kvalitet til mange noder i stedet for en tjeneste med høyere kvalitet til et færre antall noder.

Adaptivitet i forbindelse med kontinuerlige medier kan oppnås på flere måter. Den enkleste måten er å lagre flere versjoner av en ressurs (for eksempel en video), hvor hver versjon har ulik kvalitet. Gitt at en mottakernode, N_m, ønsker å motta en ressurs, r, fra en annen node på nettverket, N_r. Båndbredden til r kan da betegnes som $B(r)$, tilgjengelig inn-båndbrede for noden N_m kan betegnes som $B_{inn}(N_m)$, og tilgjengelig ut-båndbredde for N_r kan betegnes som $B_{ut}(N_r)$. For et objekt som skal streames må følgende ulikhet være tilfredsstilt: $B(r) \leq min(B_{inn}(N_m), B_{ut}(N_r))$. Fordi den tilgjengelige båndbredden både hos ressurs- og mottakernoden kan variere, blant annet på grunn av nye mottakere eller andre variasjoner i nettverket, kan det bli nødvendig å bytte objekt underveis i leveransen. Det største problemet med denne løsningen er at det er nødvendig å lagre flere versjoner av samme objekt, noe som fører til større krav til lagringskapasitet per objekt. I tillegg kan altså leverandørnoden bli nødt til å bytte versjon underveis i leveransen etter hvert som leveransebetingelsene endrer seg.

4.3.1 Lagdeling

En annen løsning for å oppnå adaptivitet er å dele et medieobjekt, for eksempel video, i flere lag. Lagdeling i denne sammenheng betyr å dele et medieobjekt i flere deler. Ved å kombinere samtlige lag oppnås det opprinnelige objektet. En delmengde av lagene kan også spilles av i kombinasjon. Jo færre lag som spilles av, desto mer blir kvaliteten på avspillingen redusert. Dermed kan en ressurskilde justere antall lag som spilles av etter verdiene av $B_{ut}(N_r)$ og $B_{inn}(N_m)$. Figur 4.8 illustrerer prinsippet med lagdeling.



Figur 4.8: Lagdeling av en mediestrøm.

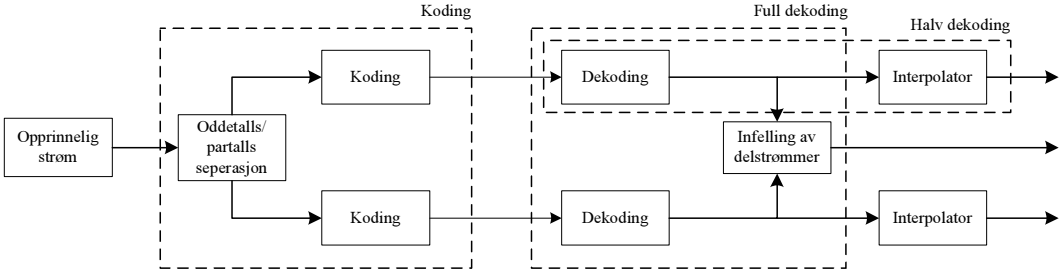
I figuren er et medieobjekt delt i n delstrømmer som sammensatt danner én mediestrøm. For å oppnå adaptivitet må færre enn n delstrømmer kunne brukes for å danne en sammensatt strøm.

For video er det flere metoder som gjør det mulig å lage flere delstrømmer av én opprinnelig strøm. [Briceño et al., 1999] har foreslått en metode som betrakter pakker som den fundamentale enhet for overføring av video. En pakke består av en liten delmengde av pikslene i et videobilde og kan spilles av uavhengig av andre pakker. Kvaliteten på avspillingen er avhengig av hvor stor andel av pakkene som representerer et bilde som kommer fram i tide. Ved å kun sende en delmengde av pakkene, kan avsender variere $B(r)$ for å gjøre ulikheten $B(r) \leq min(B_{inn}(N_m), B_{ut}(N_r))$ sann. Denne metoden var opprinnelig ment for å løse problemet knyttet til pakketap på Internett. Ved å gjøre hver pakke uavhengig av andre pakker reduseres betydningen av pakketap. Metoden kan dermed også brukes for å tillate forsettlig pakketap (adaptiv streaming). Ulempen med pakker som er uavhengige av hverandre er at muligheten for effektiv kompresjon blir dårligere. En vanlig og effektiv kompresjonsteknikk for video er å utnytte likheter mellom etterfølgende bilder i en videostrøm. MPEG er et eksempel på dette (se kapittel 3.1). Ved å gjøre pakker og bilder uavhengige av hverandre, er det ikke i like stor grad mulig å benytte denne teknikken. I følge [Cidon et al., 1993] er kompresjonsgrad og robusthet mot pakketap generelt motstridende egenskaper for video.

For sammensatte medier, for eksempel video med tilhørende flerkanals lyd, kan de forskjellige kanalene betraktes som ulike lag i en mediestrøm. Gitt at et sammensatt medieobjekt består av én videostrøm og fem lydstrømmer, vil det være mulig å oppnå adaptivitet ved å kun sende videostrømmen og for eksempel to av lydstrømmene til mottakeren. Mottakeren kan da spille av medieobjektet som video med stereo lyd. Denne type adaptivitet forutsetter at medieobjekter lagres lagdelt, det vil si med separate delobjekter for hvert lag/kanal.

4.3.2 "Multiple Description Coding" (MDC)

Et alternativ til lagdeling er "Multiple Description Coding" (MDC). MDC var opprinnelig en metode som skulle gjøre nettverksoverføring av medier mer robust mot pakketap. Ideen var at hvis pakketap var uunngåelig, så burde medierepresentasjoner ta hensyn til dette ved å redusere graden av avhengigheter mellom pakker. Større grad av uavhengighet gjør at flere pakker kan brukes til avspilling selv om dens nabopakker ikke når fram til mottaker i tide [Goyal, 2001]. MDC muliggjør slike representasjoner ved å bryte opp avhengigheter mellom pakker i tidsdimensjonen (i motsetning til metoder med romlig lagdeling). Den enkleste formen for MDC deler en opprinnelig mediestrøm i to delstrømmer. Pakker med oddetalls sekvensnummer sendes i én delstrøm, mens pakker med partalls sekvensnummer sendes i den andre delstrømmen. Pakkene i en delstrøm kan leses uavhengig av pakkene i en annen delstrøm, men ved mottak må strømmene kombineres for å oppnå full rekonstruksjon av medieobjektet. Figur 4.9 viser prinsippet med MDC i et tilfelle hvor to delstrømmer brukes.



Figur 4.9: MDC med to delstrømmer [Goyal, 2001].

I figuren deles først den opprinnelige strømmen i to strømmer. Hver delstrøm kodes så til et passende format før det streames over nettverket. Hos mottaker dekodes strømmene, enten ved full dekoding (den opprinnelige strømmen gjenopprettes) eller ved halv dekoding. Ved halv dekoding brukes kun den ene delstrømmen for å generere en dekodet og avspillbar strøm. Halv dekoding, eller delvis dekoding generelt, kan brukes for å oppnå adaptivitet. Siden hver delstrøm kan spilles av uavhengig av andre strømmer, kan sender velge å kun sende en delmengde av delstrømmene. Siden MDC lager delstrømmer ved å separere etterfølgende pakker (i tidsdimensjonen), vil et slikt pakketap medføre "hull" i strømmen. Disse hullene må fylles ved å interpolere/konstruere de manglende pakkene. Resultatet blir ikke like godt som den opprinnelige strømmen, men kan være en god tilnærming. En strøm kan deles i et vilkårlig antall delstrømmer, og graden av adaptivitet kan derfor varieres etter behov. Jo flere delstrømmer, desto mindre kompresjon kan gjøres på grunnlag av likhet mellom

etterfølgende pakker i hver delstrøm. Derfor er det også med denne metoden en avveiing mellom grad av kompresjon i forhold til adaptivitet [Goyal, 2001].

4.4 Tjenestekvalitet

Som beskrevet i kapittel 2 er et P2P-system et distribuert system som i utgangspunktet ikke har sentrale enheter som alltid er til stede. Nodene i et P2P-system kan befinne seg i vidt forskjellige nettverk, og kan melde seg inn og ut av systemet når som helst. I tillegg er det nodene selv som bidrar med ressurser og tjenester. Disse egenskapene gjør et P2P-system både dynamisk, ustabilt og heterogent.

At et P2P-system er *dynamisk* vil si at nettverkstopologien er i stadig endring og at tilgjengelige ressurser og tjenester varierer. Disse variasjonene gjør at ressurser og tjenester kan bli *ustabile*, blant annet fordi de kan forsvinne når som helst. I tillegg vil den underliggende nettverksinfrastrukturen ofte være ustabil til tross for transportprotokoller som TCP eller RTP. Med *heterogent* menes at nettverket består av mange noder med vidt forskjellige egenskaper. Nodene kan befinne seg i ulike endenettverk med forskjellig båndbredde og responstid. I tillegg er ofte nodene ulike, og har vidt forskjellige egenskaper i forhold til prosessering, lagring og kommunikasjon. Disse egenskapene bidrar til utfordringer knyttet til å tilby tjenestekvalitet i P2P-systemer generelt — uavhengig av om nettverkslag under applikasjonslaget har mekanismer som legger til rette for det.

4.4.1 Parametere og utfordringer

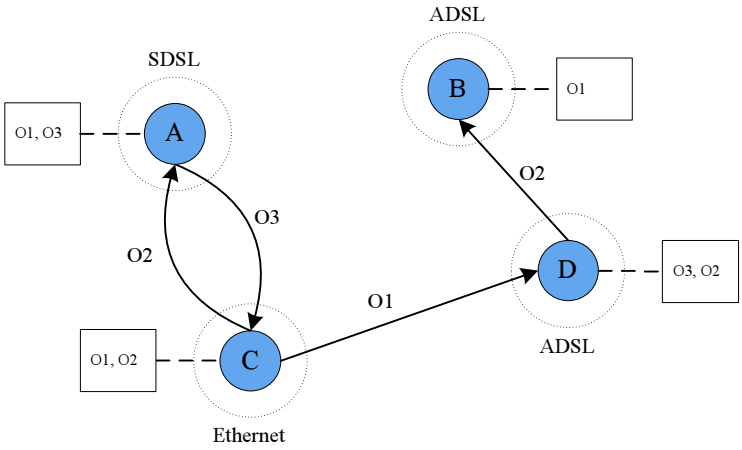
Streaming i et P2P-system skaper ytterligere utfordringer ved sikring av tjenestekvalitet sammenlignet med tjenester som verken har sanntidskarakteristikker eller krever kontinuerlig leveranse over lengre tidsperioder. Utfordringer som er direkte knyttet til streaming kan til dels løses i nettverkslaget ved at spesielle protokoller (RTP/RTCP) brukes. Mange problemer skyldes imidlertid ikke streaming alene, men kombinasjonen av P2P og streaming. Disse problemene må løses i applikasjonslaget.

For å kunne spesifisere hvilke utfordringer som er knyttet til tjenestekvalitet ved streaming i P2P-nettverk, er det nødvendig å definere hva som regnes som viktige kvalitetsparametere. Følgende parametere har stor betydning for tjenestekvalitet:

- **Gjenfinningsgrad:** At alle ressurser og replikater av en ressurs kan gjenfinnes til enhver tid.

- **Pakketap/tapte tidsfrister:** Hvor mange pakker som når fram til mottaker i tide.
- **Tilgjengelighet:** At en ressurs ikke blir utilgjengelig mens den er i bruk.

Figur 4.10 viser et eksempelscenario hvor forskjellige objekter streames mellom noder i et P2P-system. En heltrukken pil symboliserer en strøm mellom to noder. Nodene befinner seg i hvert sitt endenett av ulik type. Node C befinner seg eksempelvis i et Ethernet, node A i et symmetrisk DSL-nett mens nodene B og D befinner seg i et asymmetrisk DSL-nett. I dette eksempelet er samtlige noder unntatt B både leverandør og konsument. Node B er kun konsument, men disse forholdene kan raskt kunne endre seg ved nye objektforespørsler. Hver node har et antall objekter som den deler med resten av nettverket. Tilgjengelige objekter i nettverket er O1, O2 og O3. Flere av objektene er replikert, det vil si finnes i flere kopier hos ulike noder.



Figur 4.10: Eksempelscenario — streaming i P2P-nettverk.

Flere problemer som påvirker tjenestekvalitet kan oppstå i dette scenariet. Gitt at node C melder seg ut av nettverket mens den leverer strømmer til A og D. I et slikt tilfelle vil tilgjengeligheten til objektene O1 og O2 bli rammet. Den eneste løsningen vil da være å finne alternative noder som kan levere de samme objektene. Dette forutsetter imidlertid at det er mulig å gjenfinne nødvendige replikater. Hvis det ikke er mulig å gjenfinne alle replikater av et objekt kan konsekvensen bli at et objekt blir utilgjengelig — selv om det fortsatt finnes i nettverket.

At noder befinner seg i ulike endenettverk er heller ikke problemfritt. For at et P2P-nettverk skal fungere, er det nødvendig at det er balanse mellom tilgjengelige og konsumerte ressurser. Endenettverk av ulik type kan være en utfordring

i denne sammenheng. En løsning på dette problemet er adaptiv streaming. Et objekt som skal streames mellom to noder må ikke ha båndbreddekrav høyere enn ut-båndbredden til sendernoden eller inn-båndbredden til mottakernoden. Men selv med adaptivitet kan problemer oppstå. Variasjoner i båndbredde eller forsinkelse i nettverk er et vanlig problem. Dersom nettverket ikke er i stand til å dynamisk justere båndbredden på objektet som skal sendes, kan avbrudd oppstå. Dette vil ramme den opplevde tjenestekvaliteten.

Utfordringer knyttet til streaming i P2P-nettverk kan oppsummeres som følger:

- **Gjenfinning:** Alle replikater må kunne gjenfinnes slik at det er mulig å opprette alternative strømkanaler hvis et objekt blir utilgjengelig.
- **Tilgjengelighet:** Konsekvensene av at en node (og dermed dets objekter) blir utilgjengelig må begrenses.
- **Heterogenitet:** Et P2P-nettverk må fungere selv om noder befinner seg i ulike typer nettverk.
- **Ustabilitet:** Variasjoner i båndbredde og forsinkelse må i så liten grad som mulig føre til avbrudd i leveransen.

Gjenfinning og ruting av forespørsler er beskrevet i kapittel 2.6. Det finnes både sentraliserte og desentraliserte løsninger som garanterer at et objekt i nettverket blir funnet og rutet til. Disse løsningene kan også finne alle replikater av et objekt. Gjenfinning er altså et løsbart problem, og vil ikke bli diskutert videre i denne rapporten. Tilgjengeligheten til et objekt er åpenbart viktig når det skal streames og spilles av i sanntid hos en annen node. I P2P-nettverk, hvor noder kan falle ut av nettverket når som helst, er utilgjengelighet et spesielt aktuelt problem. Sammen med ustabilitet utgjør utilgjengelighet kanskje den største trusselen mot tjenestekvalitet i P2P-nettverk. Når endenettverkene i tillegg har vidt forskjellige egenskaper (blant annet båndbredde og forsinkelse), blir det en enda større utfordring å oppnå kontinuerlig leveranse.

4.4.2 Kontroll/Administrasjon

Fordi et P2P-nettverk i utgangspunktet kan bestå av alle typer noder på randen av Internett er det umulig for P2P-systemet å administrere lavere nettverkslag (lavere enn transportlaget). Det er ikke mulig å reservere ressurser i nettverket, og det er også vanskelig å forutse flaskehalser i og mellom ulike nettverk. P2P-systemer må derfor levere tjenester etter "best-effort"-prinsippet, det vil si å gjøre så mye som mulig for å levere en pålitelig tjeneste uten å kunne gi garantier (verken deterministiske eller statistiske). For at "best-effort" skal bli

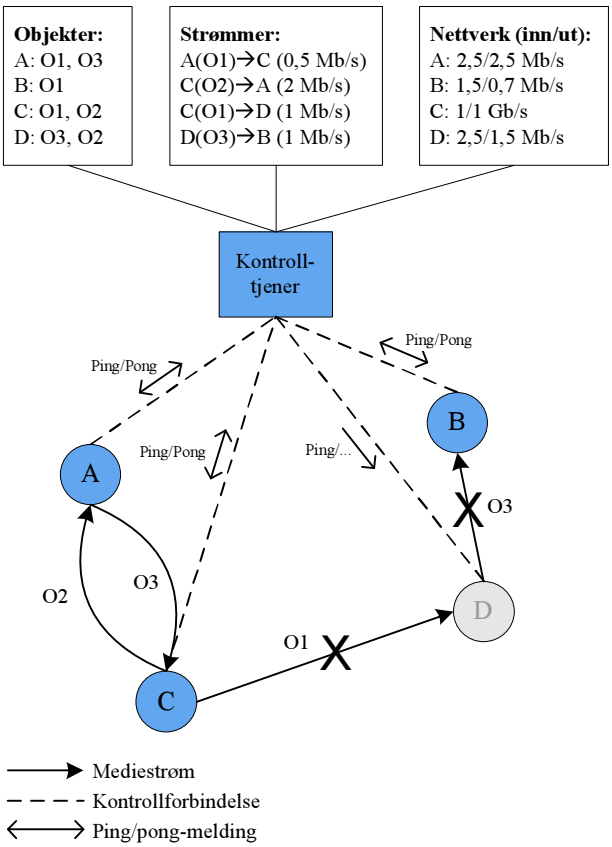
så bra som mulig, er det ønskelig med en viss kontroll over ressurser og aktiviteter i nettverket. Et dynamisk nettverk med ustabile nettverksomgivelser og varierende ressurstilgjengelighet må være følsomt for å kunne reagere raskt overfor endringer som kan medføre avbrudd i en strøm. Dette krever en form for administrasjon/overvåkning som kan sette i verk nødvendige tiltak når noe uforutsett skjer.

Når et P2P-system skal administreres (ressurser/objekter, strømmer og tilgjengelig båndbredde skal overvåkes og tiltak skal iverksettes hvis noe skjer), er det to alternative modeller som kan brukes. Som ved ressurslokalisering og ruting står valget mellom en hybrid- eller en ekte P2P-modell. En hybrid løsning vil inkludere en sentralisert tjener og betyr altså sentralisert kontroll. En ekte P2P-modell gir mer autonomi/selvstyre til hver enkelt node ved at samtlige noder deltar i administrasjonen av nettverket. I prinsippet må altså en av følgende modeller brukes:

- Sentralisert kontroll (hybrid løsning).
- Selvstyrt kontroll (ekte P2P).

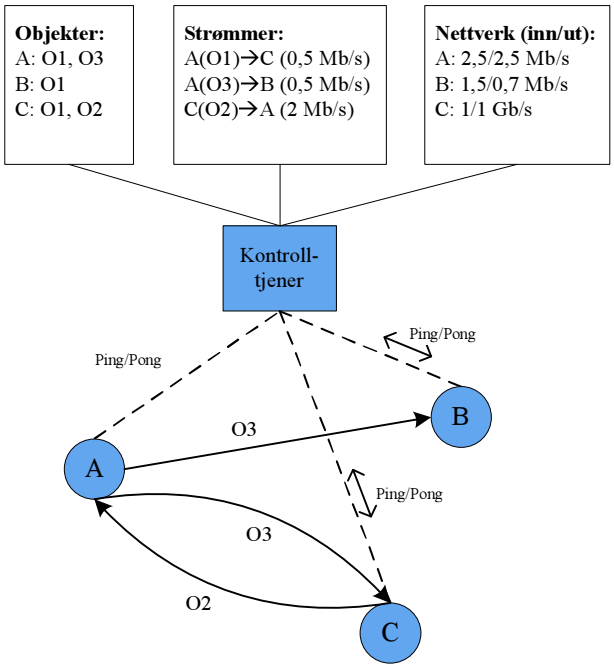
Det er fordeler og ulemper med hver modell. En sentralisert løsning samler all informasjon om ressurser, strømmer og hver nodes nettverkskapabiliteter i én sentral tjener. Denne tjeneren må stadig holde kontroll over hvilke noder som er med i nettverket og hvilke data som sendes mellom hvilke noder. En veldig enkel sentralisert løsning er illustrert i figur 4.11. I denne løsningen sender tjeneren "ping"-meldinger til hver node ved jevne mellomrom. Så lenge noden er aktiv, sender den tilbake en "pong"-melding til tjeneren. Tjeneren vet da at noden er aktiv og dermed at den ikke har forlatt nettverket. Sammen med "pong"-meldingen kan noden sende annen informasjon, for eksempel forsinkelsen på strømmen den sender eller mottar. Tjeneren kan bruke denne informasjonen for å foreta avgjørelser om eventuelle endringer som må utføres (annen kilde for strømmen, justering av adaptivitet osv.).

Figur 4.11 viser samme situasjon som i figur 4.10 bortsett fra at node D har falt ut. Dette medfører at D verken er i stand til å levere objektet O3 til node B, eller kan motta strømmen med objekt O1 fra node C. Kontrolltjeneren oppdager at node D har falt ut ved at den ikke svarer på "ping"-meldingen. For å unngå at node B opplever brudd på strømmen, må kontrolltjeneren søke etter alternative steder å hente objekter fra.



Figur 4.11: Sentralisert kontroll — node faller ut.

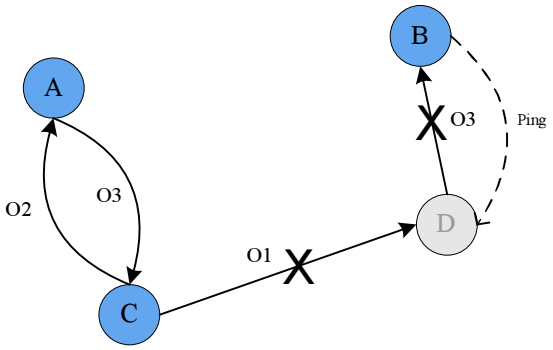
I denne situasjonen har node A et replikat av objektet O3. I tillegg har A tilstrekkelig ledig nettverkskapasitet til at den kan overta leveransen av objektet. Kontrolltjeneren gir derfor A beskjed om at den skal opprette en strøm med O3 til B. Den nye situasjonen er illustrert i figur 4.12.



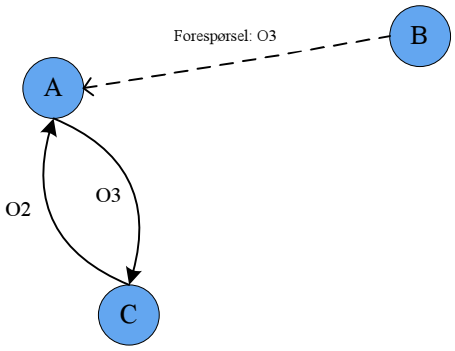
Figur 4.12: Sentralisert kontroll — ny node overtar.

Selvstyrt kontroll gir deltakerne i P2P-nettverket et felles ansvar for å utføre administrasjon/overvåking av objekter, strømmer og tilgjengelig båndbredde. En slik løsning distribuerer kontrollen mellom nodene som sammen må håndtere unntakssituasjoner som oppstår. Et slikt system kan organiseres på tilsvarende måte som et fullt desentralisert overlagsnettverk. I tillegg til ressurslokalisering og ruting kan da overlagsnettverket brukes til strømovervåking og annen nettverksadministrasjon.

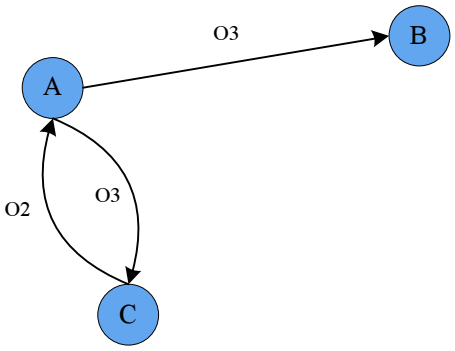
Figur 4.13 viser samme scenario som tidligere, men med en administrasjonsløsning basert på selvstyrt kontroll. I dette eksempelet brukes et overlagsnettverk som lokaliserer samtlige tilgjengelige ressurser i nettverket. Ved brudd kan en node lokalisere et annet replikat av objektet ved å sende en spørring til overlagsnettverket. Figur 4.13(a) viser situasjonen rett etter at node D har falt ut av nettverket. Strømmen til node B har blitt brutt, og B får heller ingen respons når den forsøker å sende en "ping"-melding til D. Node D bruker da overlagsnettverket for å lokalisere eventuelle replikater av objektet den har mottatt. Node A har et slikt replikat, og B sender en objektforespørsel til A (figur 4.13(b)). Siden node A i dette tilfellet har tilstrekkelige ressurser til å levere objektet til B, opprettes det en strøm mellom de to nodene (figur 4.13(c)).



(a) Node har falt ut.



(b) Ny strømforespørsel.



(c) Ny strøm etablert.

Figur 4.13: Selvstyrt/desentralisert kontroll — virkemåte.

I dette eksempelet er det hver enkel node selv som håndterer feilsituasjoner. Overlagsnettverket brukes for å lokalisere alternative ressurskilder, og noden sender selv forespørsel etter en ressurs. Dette er en veldig enkel løsning, men illustrerer prinsippet med autonomi i forhold til sentral kontroll.

Fordeler med sentralisert løsning:

- Sentralisering er enkelt. Løsningen krever blant annet lite utveksling av informasjon mellom noder i nettverket.
- En sentral tjener vil kjenne til samtlige noder i nettverket og kan derfor foreta avgjørelser basert på fullstendig informasjonsgrunnlag. Dette kan gi mer optimale løsninger totalt sett enn om informasjonsgrunnlaget hadde vært mer begrenset.
- En sentral tjener kan foreta avgjørelse raskt fordi den allerede besitter nødvendig informasjon og slipper å hente den fra andre noder først.

Ulemper med sentralisert løsning:

- En sentral tjener kan bli en flaskehals.
- Sentrale enheter gjør et system mindre skalerbart.
- Hvis tjeneren faller ut vil hele systemet rammes.

Fordeler med selvstyre/desentralisering:

- Desentralisert administrasjon har ingen sentrale enheter som utgjør en flaskehals, begrenser skalerbarheten eller er vesentlig for at systemet skal fungere.
- Nodene kan ta avgjørelser på vegne av seg selv og kan derfor reagere raskt på endringer som ikke krever informasjon om andre noder.

Ulemper med selvstyre/desentralisering:

- Mange kontrollmeldinger må sendes mellom noder i P2P-nettverket.
- Det kreves mange meldinger for å innhente tilstrekkelig informasjon slik at det er mulig å foreta optimale løsninger ved feil.
- Mange meldinger kan føre til at det kan ta lengre tid å foreta en avgjørelse ved feil enn ved sentralisert administrasjon.

Det er også mulig å tenke seg løsninger som både har en viss grad av selvstyre og sentralisert kontroll. En sentral enhet vil da ta avgjørelser som en enkelt node ikke kan foreta uten å måtte innhente informasjon fra mange andre noder. Hver enkelt node kan foreta avgjørelser på vegne av seg selv og noden den sender en strøm til eller mottar en strøm fra, blant annet hvilken grad av adaptivitet som trengs for en gitt strøm.

Kapittel 5

Multinodestreaming

Når dataleveranse kan skje i stabile nettverk hvor noder aldri eller sjelden blir utilgjengelige, vil et P2P-system hvor én node har ansvaret for å levere en strøm fungere. Utfordringer knyttet til heterogene nettverk kan løses ved å benytte adaptiv streaming, og skulle en feil oppstå fra tid til annen kan dette aksepteres. En slik ideell situasjon er imidlertid ikke tilfelle på Internett. I P2P-nettverk hvor deltakende noder er vanlige privateide datamaskiner lokalisert rundt om i verden, vil det være stor sannsynlighet for at en node melder seg ut av nettverket eller blir utilgjengelig av andre grunner. Store avstander på nettverket, og varierende kvalitet på tjenestene som leveres av ulike Internett-tilbydere, skaper også utfordringer i form av uforutsigbarhet og ustabilitet. I en slik situasjon er ikke nødvendigvis et system hvor én node leverer en hel strøm tilfredsstillende fordi det vil medføre avbrutt leveranse og dårlig tjenestekvalitet. I dette kapittelet vil vi derfor foreslå en metode som har som mål å tilby stabile tjenester med høyere grad av tjenestekvalitet, til tross for at den skal operere i ustabile omgivelser. Både utfordringer som ustabile nettverksomgivelser og stadig varierende nettverkstopologi skal kunne håndteres med denne metoden. Vi har valgt å kalle metoden for multinodestreaming fordi den benytter flere noder for å streame et objekt.

5.1 Introduksjon

Utfordringene knyttet til å levere data i P2P-nettverk er mange. Jo lenger tid leveransen tar, desto større sannsynlighet er det for at noe kan skje som enten ødelegger leveransen fullstendig eller degraderer kvaliteten på leveransen. Ved vanlig filoverføring kan slike utfordringer løses ved at TCP-protokollen sender tapte data på nytt. Dermed vil det mottatte objektet kunne leses som om at nettverksoverføringen hadde vært feilfri. Ved leveranse av sanntidsdata som video og lyd, er ikke alltid slike løsninger tilfredsstillende fordi data som sen-

des på nytt ankommer for sent til å kunne spilles av. Ved filoverføring er det mulig å gjenoppta leveransen om avsendernoden blir utilgjengelig. Ved sanntidsleveranse vil hele sanntidsaspektet forsvinne om det blir leveransebrudd. Utfordringen ligger i å sikre at leveransebrudd skjer så sjelden som mulig i tillegg til at kvaliteten på leveransen til enhver tid maksimeres.

5.1.1 Idé: Motta delstrømmer fra flere noder

Å sikre ressurstilgjengelighet er en stor utfordring i P2P-nettverk fordi noder uten varsel kan forsvinne fra nettverket. Konsekvensen av at en node forsvinner er at dens ressurser blir utilgjengelige. Noder som bruker ressursene til en annen node som forsvinner vil oppleve at ressursen blir utilgjengelig. Med en modell hvor en node henter en ressurs fra én node alene, er ressurstilgjengelighet direkte knyttet til nodetilgjengelighet. Hvis noden blir utilgjengelig, blir også ressursen utilgjengelig inntil en ny node med tilsvarende ressurs blir lokalisert og en ny forbindelse opprettes. For tjenester som leveres i sanntid, for eksempel streaming av digitale medier, vil dette medføre strømavbrudd og dermed avbrudd i avspillingen av mediet.

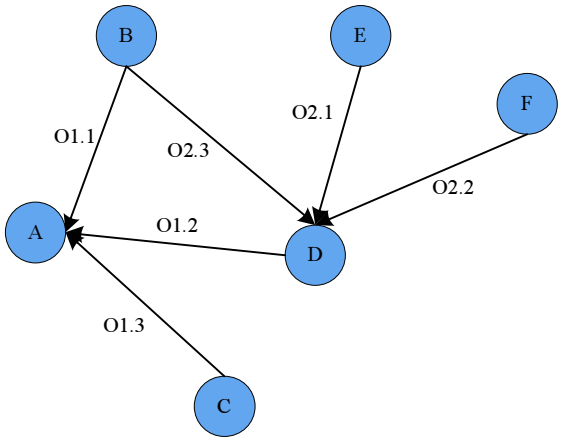
I kapittel 4.3 ble metoder for å oppnå adaptivitet beskrevet. Adaptivitet kan blant annet oppnås ved at et objekt deles i flere delobjekter, og hvor kun en delmengde av delobjektene streames til mottakeren. Ved å ikke sende alle delobjektene degraderes kvaliteten på avspillingen av objektet, men dette forhindrer ikke avspillingen i seg selv. Hvert delobjekt kan altså spilles av uavhengig av andre delobjekter, og jo flere delobjekter som spilles av i parallell, desto bedre kvalitet blir det på avspillingen. Fordi det i P2P-nettverk ofte er mange replikater av et objekt, vil det også være flere replikater av hvert delobjekt. I dette kapittelet vil vi foreslå og spesifisere en metode som utnytter denne egenskapen for å øke datatilgjengeligheten i P2P-nettverk. Metoden tar ikke sikte på å øke nodetilgjengeligheten, men bryter i stedet opp det ellers nære forholdet mellom node- og ressurstilgjengelighet ved å spre ansvaret for å levere et objekt mellom flere noder. Med denne metoden får flere noder ansvaret for å levere ett objekt ved at ulike noder leverer ulike delobjekter. Hvis en node faller ut vil kun en delmengde av delobjektene bli utilgjengelige, og avspillingen hos mottakernoden kan fortsette med degradert kvalitet (i stedet for å bli fullstendig avbrutt) inntil et replikat er lokalisert.

Gitt et scenario hvor nodene A, B, C, D, E og F deltar i et P2P-nettverk hvor multinodestreaming benyttes. I scenariet er det objektforespørsler for objektene O1 og O2, og hvert objekt deles i tre delobjekter. Tabell 5.1 gir en oversikt over hvilke noder som kan levere de forskjellige objektene. De objektene som en node mottar er angitt i parentes.

Node	Objekter
A	(O1)
B	O1, O2
C	O1
D	O1, (O2)
E	O2
F	O2

Tabell 5.1: Oversikt over noder og tilgjengelige objekter.

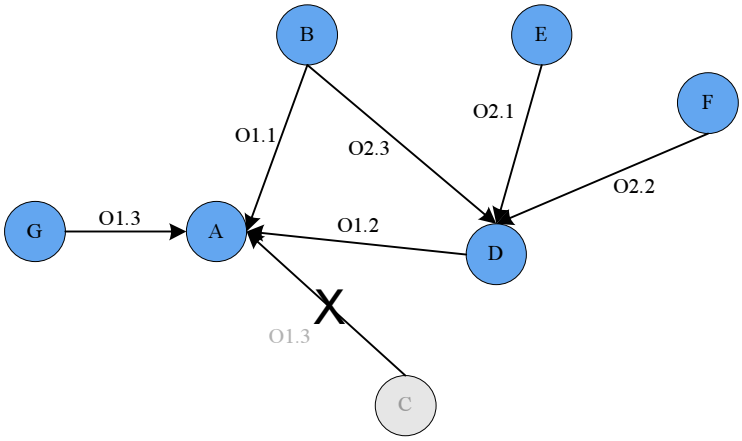
Dette scenariet kan illustreres som i figur 5.1. Her deles ansvaret for å levere et objekt mellom tre noder. Hvis en node faller ut, vil kvaliteten på avspillingen kun degraderes med en tredjedel. Hvis objektet finnes i flere uutnyttede og ledige replikater, kan full kvalitet gjenopprettes ved å forespørre noden som er i besittelse av dette replikatet. Full avspillingskvalitet kan da gjenopprettes etter en tid hvis denne noden har kapasitet til å levere delobjektet.



Figur 5.1: Multinodestreaming med tredelte objekter.

En tjeneste hvor brudd kun medfører kvalitetsdegradering av avspillingen vil kunne oppleves som en bedre tjeneste av brukeren enn om hele avspillingen ble brutt. Jo flere deler et objekt deles i, desto større grad av ansvarsdeling, og dermed også tilgjengelighet, vil kunne oppnås. Som beskrevet i kapittel 4.3, vil det alltid være en avveiing mellom kompresjon og delingsfaktor for et objekt. Tilgjengeligheten økes dermed på bekostning av den totale størrelsen til objektet, noe som er en ulempe og et potensielt problem med metoden. Ressursene som kreves for å levere et delobjekt vil imidlertid være mindre enn om hele objektet skulle leveres, og metoden bidrar således også til å spre lasten forbundet med å levere et objekt over flere noder.

Tjenestekvalitet er, som tidligere nevnt, blant annet avhengig av datatilgjengelighet. Det er imidlertid en forskjell mellom faktisk datatilgjengelighet og opplevd datatilgjengelighet ved avspilling. Gitt et scenario tilsvarende det i figur 5.1 hvor det i tillegg finnes en ekstra node, G, som også har objektet O1. Dette scenariet er illustrert i figur 5.2.



Figur 5.2: Multinodestreaming med overtakelse av delstrøm.

Også i dette scenariet deles et objekt i tre delobjekter. I utgangspunktet leverer ikke G data til noen andre noder. Hvis for eksempel node C faller ut av nettverket, vil node G kunne overta ansvaret for delstrømmen som ikke lenger blir sendt av C. Den faktiske datatilgjengeligheten er dermed fortsatt 100 % fordi det er et tilstrekkelig antall noder som er i stand til å levere det antall delobjekter som på det tidspunktet er forespurt. Den opplevde datatilgjengeligheten er imidlertid ikke 100 % i den tiden det tar fra node C faller ut til node G har overtatt ansvaret for å streame delobjekt O1.3 og node A mottar strømmen. Opplevd datatilgjengelighet er en viktig faktor som påvirker tjenestekvaliteten for slike tjenester. Vi vil derfor også foreslå en metode som gjør gapet mellom faktisk og opplevd datatilgjengelighet så liten som mulig, samtidig som nodene i nettverket ikke belastes unødig mye. Multinodestreaming har altså to aspekter:

- Dele ansvaret for å levere en strøm mellom flere noder.
- Minimere gapet mellom faktisk og opplevd datatilgjengelighet.

I det følgende delkapittelet vil vi se videre på hvordan et P2P-system som skal levere digitale medier i sanntid kan bygges opp. De funksjonelle egenskapene som er sentrale for å oppnå høyere grad av tilgjengelighet vil blant annet bli diskutert nærmere. Videre vil vår foreslåtte metode bli spesifisert nærmere

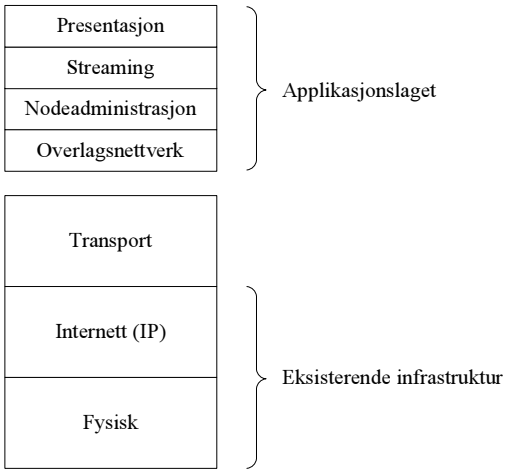
med vekt på å beskrive funksjonelle krav, designprinsipper og metoden i seg selv.

5.1.2 Virkemåte

Multinodestreaming er på ingen måte knyttet til en spesifikk applikasjonstype utover at metoden naturligvis er sterkt knyttet til leveranse av kontinuerlige digitale medier (eksempelvis video og lyd). I den følgende diskusjonen vil vi imidlertid bruke et mer konkret totalsystem som rammeverk for diskusjonen rundt og utformingen av multinodestreaming-metoden.

I nettverksmodellen som tidligere har vært presentert (blant annet i figur 2.7), vil muligheten for parallell streaming fra flere noder muliggjøres i applikasjonslaget. Applikasjonslaget vil bestå av flere komponenter som er avhengige av hverandre. Et overlagsnettverk vil være nødvendig for å forespørre ressurser og rute en forespørsel til alle noder som har et replikat av en bestemt ressurs. I tillegg vil det være nødvendig med funksjonalitet for å generere og spille av strømmer. Denne funksjonaliteten vil implementeres i et streaminglag. Presentasjon av data (for eksempel avspilling av en video) vil gjøres i et presentasjonslag.

Med enkeltnodestreaming vil i prinsippet overlagsnettverket, streaminglaget og presentasjonslaget være tilstrekkelige for å lage en P2P-basert streamingtjeneste. Med multinodestreaming trengs imidlertid også et lag for å administrere noder og delstrømmer. Dette laget plasseres mellom overlagsnettverket og streaming fordi det er avhengig av overlagsnettverket og leverer tjenester til streaminglaget. Figur 5.3 illustrerer denne lagdelingen. Nodeadministrasjonslaget bruker overlagsnettverket for å finne og holde orden på de forskjellige nodene i nettverket som er i besittelse av en bestemt ressurs. Skulle en node falle ut slik at en delstrøm forsvinner, må nodeadministrasjonslaget benytte overlagsnettverket til å lokalisere alternative replikater. Streaminglaget for multinodestreaming vil være tilnærmet det samme som streaminglaget i en modell hvor enkeltnodestreaming benyttes. Den eneste forskjellen er at streaminglaget med multinodestreaming i tillegg må ha funksjonalitet for å synkronisere delstrømmer.



Figur 5.3: Nodeadministrasjon sin plass i applikasjonslaget.

Videre i dette kapittelet vil vi blant annet diskutere hvordan multinodestreaming bør implementeres for å gjøre gapet mellom faktisk og opplevd datatilgjengelighet minst mulig. Fokus vil derfor være rettet mot hvordan nodeadministrasjonslaget bør bygges opp og hvilke funksjoner det må implementere. Under- og overliggende lag (overlagsnettverket og streaminglaget) er ikke direkte delaktige i metoden, og vil derfor heller ikke bli omtalt eksplisitt. Disse lagene er imidlertid nødvendige for et totalsystem, men har bare en indirekte virkning på implementasjonen av nodeadministrasjonslaget.

Funksjoner

Funksjonsmessig vil et fullstendig P2P-basert system for streaming av digitale medier måtte ha en rekke funksjoner ut over muligheten til å hente et objekt fra flere noder i parallell. Disse funksjonene tilbys av under- og overliggende lag i applikasjonen og nettverket, men er viktig å ta i betraktning ved utforming av delsystemet som tar seg av nodeadministrasjon. I tillegg skal et slikt system operere i bestemte omgivelser. Omgivelsene påvirker hvordan systemet må utformes, og er derfor også viktige.

Funksjonelt sett må et P2P-basert system for streaming av digitale medier med mulighet for multinodestreaming ha følgende egenskaper:

1. **Søk:** Det må være mulig for en bruker å søke etter en ressurs, enten ved hjelp av stikkord (hvis det er metadata tilknyttet ressurser) eller ved hjelp av ressursens identifikator.

2. **Gjenfinning:** Systemet må være i stand til å søke etter og gjenfinne alle replikater av en bestemt ressurs.

3. **Resultatevaluering og forhandling**: Når alle replikater av en ressurs er funnet, må mottakernoden vurdere hvilke noder som er i besittelse av ressursen. Når dette er gjort må den innlede forhandlinger med de høyest rangerte nodene.

4. **Ruting:** Systemet må være i stand til å lokalisere noder basert på interne identifikatorer og å rute ressursforespørsler til disse.

5. **Streaming:** Et medieobjekt må kunne sendes over nettverket som en strøm. Det må også være mulig å motta strømmer og å spille av disse.

6. **Synkronisering og bufring:** Mottaker av en strøm må være i stand til å bufre og synkronisere delstrømmer slik at de kan spilles av i parallell selv om de mottas med en viss tidsforskyvning.

7. **Strømadministrasjon:** Overvåking av strømmer og iverksettelse av tiltak hvis en node blir utilgjengelig er essensielt for å kunne tilby høy opplevd datatilgjengelighet. Strømadministrasjonen må reagere raskt for å tilpasse seg endringer i nettverket.

8. **Tilgangskontroll:** En node må være i stand til å vurdere om den kan akseptere en ressursforespørsel. Dette innebærer blant annet å vurdere om den har tilstrekkelige ressurser ledig til å utføre leveranse av data.

Funksjonene omtalt i punkt 1, 2 og 4 implementeres i overlagsnettverket. Punkt 5 og 6 er knyttet til streaminglaget og implementeres ikke direkte i multinodestreaming-metoden. At funksjonene i punkt 3, 7 og 8 blir ivaretatt er essensielt for at opplevd datatilgjengelighet blir optimal. Derfor vil disse punktene inngå i nodeadministrasjonslaget og vil bli videre omtalt de kommende delkapitlene.

Omgivelser

Omgivelser i denne sammenheng regnes som eksterne systemer, nettverk og aktører som et system er avhengig av eller påvirkes av på en eller flere måter. For et P2P-basert system for streaming av digitale medier, er omgivelsene svært komplekse og uoversiktlige. I tillegg er de i svært liten grad konfigurerbare for P2P-systemet. Internett er kanskje den mest kompliserte og også viktigste delen av omgivelsene til et P2P-system. I tillegg regnes selve maskinene som deltakernodene kjører på som en del av et P2P-system sine omgivelser. Internett kjennetegnes ved at det består av et stort antall mindre nettverk som kommuniserer over felles stamnett. De ulike delnettverkene kan ha svært ulike karakteristikker og egenskaper, hvor asymmetrisk båndbredde og stor variasjon i responstider er blant utfordringene et P2P-system må håndtere.

Maskinene som er tilkoblet endenettverkene på Internett er også svært ulike. Både prosessor- og lagringskapasitet kan variere, i tillegg til oppetid og andre faktorer som er knyttet til maskinens grensesnitt til nettverket.

Omgivelsene til et P2P-system er altså preget av uorden og usikre faktorer som det er vanskelig å påvirke. Tilgjengeligheten påvirkes også av disse faktorene, noe som gjør at det er spesielt viktig å ta hensyn til omgivelsene ved design av et robust P2P-system. Ved design av et P2P-system må man ta utgangspunkt i at omgivelsene ikke kan manipuleres i særlig grad. I stedet må P2P-systemet designes slik at det er tilpasningsdyktig og fleksibelt slik at det er i stand til å operere under skiftende forutsetninger.

5.1.3 Målsetning

Målet med det videre arbeidet er å rette fokus på hvordan den opplevde tilgjengeligheten kan forbedres ved streaming av digitale medier i P2P-nettverk. Vi ønsker å vurdere om det finnes løsninger som kan bidra til ressurstilgjengelighet uten å bryte med viktige designprinsipper knyttet til P2P-arkitekturen. Vi har allerede antydet at en slik løsning kan ta utgangspunkt i multinodestreaming. Med dette utgangspunktet ønsker vi å vurdere hvordan et fullstendig system kan designes. Videre vil vi konsentrere diskusjonen rundt funksjonaliteten i nodeadministrasjonslaget.

Mer konkret ønsker vi videre å:

- Fastsette hvilke egenskaper et totalsystem må implementere.
- Slå fast hvilke designprinsipper som skal legges til grunn for utformingen av et system som ivaretar tilgjengelighetsproblematikken ved å implementere et nodeadministrasjonslag.
- Foreslå et overordnet design av et fullstendig P2P-system med fokus på å identifisere moduler (som representerer forskjellige hovedfunksjoner) og relasjonene mellom disse.
- Foreslå en metode som bidrar til å gjøre den opplevde ressurstilgjengeligheten i et P2P-system så høy som mulig gitt de forutsetningene som er gitt av omgivelsene (usikre nettverk, uforutsigbar nodetilgjengelighet osv.). Beskrivelsen av denne metoden vil blant annet inkludere spesifikasjon av tilstander, meldinger og prosesser.

Med vårt bidrag ønsker vi å rette fokus på streamingtjenester i P2P-nettverk ved å se på om, og eventuelt hvordan, slike systemer kan gjøres pålitelige og robuste. Fokus på tjenestekvalitet bør være sentralt ved design av slike systemer fordi det er avgjørende at et system kan levere tjenester som tilfredsstiller

brukernes behov og krav. Tilgjengelighet er åpenbart en viktig parameter i denne sammenheng, og ved å fokusere på og forsøke å finne løsninger orientert rundt denne egenskapen, kan vi bidra ett skritt i retningen av P2P-baserte systemer som også kan streame mediedata.

5.2 Systembeskrivelse

I dette delkapittelet vil vi beskrive systemet som implementerer multinodestreaming. Først vil vi beskrive overordnede funksjonelle egenskaper som et slikt system må implementere. Videre vil vi beskrive en designfilosofi — et sett med veiledende grunnprinsipper som ligger til grunn for viktige designmessige avgjørelser. I tillegg ønsker vi å beskrive hvordan systemet kan organiseres på modulnivå for å identifisere den logiske oppdelingen av funksjonalitet og forholdet mellom funksjoner.

5.2.1 Funksjonelle egenskaper

Et P2P-system som skal kunne levere digitale medier som video og lyd i sanntid må ha en rekke egenskaper knyttet til streaming i tillegg til selve P2P-arkitekturen. Disse egenskapene kan til en viss grad trekkes ut fra summen av generelle egenskaper ved streaming-applikasjoner og generelle egenskaper ved P2P-applikasjoner. I tillegg er det også en del egenskaper som er spesielle for *kombinasjonen* streaming og P2P. Som beskrevet i kapittel 4, og innledningsvis i dette kapittelet, er egenskapene knyttet til tjenestekvalitet spesielt viktige for slike applikasjoner.

I dette delkapittelet vil vi gi en oversikt over de funksjonelle egenskapene som vi anser som viktige for P2P-baserte streamingapplikasjoner som skal ta spesielt hensyn til ressurstilgjengelighet i uforutsigbare nettverksomgivelser. Egenskapene som presenteres i dette delkapittelet gjelder et generelt system som ikke er knyttet til en bestemt applikasjonstype utover evnen til å streame digitale medier i P2P-nettverk ved hjelp av multinodestreaming og andre generelle P2P-funksjoner (for eksempel overlagsnettverk). Vi ønsker ikke å spesifisere et system etter bestemt bruk (for eksempel fildeling) fordi vi ønsker å holde metodespesifikasjonen på et så generelt nivå at den kan anvendes i forskjellige applikasjoner. Vi trenger imidlertid et referansesystem slik at det er mulig å vurdere metoden i sammenheng med andre funksjoner og moduler. Disse egenskapene er derfor ikke å betrakte som funksjonelle krav til et bestemt system, men snarere en beskrivelse av overordnede egenskaper som et slikt system vil implementere.

De funksjonelle egenskapene til et totalsystem som skal kunne levere digitale

medier som video og lyd i sanntid kan beskrives som i tabell 5.2:

Egenskap	Beskrivelse
1	Det må være mulig å gjenfinne og lokalisere samtlige replikater av en bestemt ressurs, og å rute forespørsler til hver av disse.
2	Systemet må kunne generere en mediestrøm av et medieobjekt og sende denne over et nettverk til en bestemt node.
3	Systemet må kunne å motta flere mediestrømmer samtidig.
4	Systemet må kunne synkronisere ulike mediestrømmer i forhold til hverandre.
5	Endringer i omgivelsene som påvirker ressurstilgjengelighet skal føre til tiltak som kan opprettholde eller gjenopprette tilgjengeligheten hvis dette er mulig.

Tabell 5.2: Overordnede funksjonelle egenskaper.

5.2.2 Designfilosofi

For å spesifisere metodene for multinodestreaming vil vi ta utgangspunkt i et sett med overordnede designprinsipper som vist nedenfor. Hensikten med designprinsippene er ikke å følge dem slavisk, men heller å kunne bruke dem som en veiledende pekepinn for at metoden skal oppnå de ønskede egenskapene. Følgende prinsipper vil bli lagt til grunn for designet:

- Enkelhet
- Feiltoleranse
- Skalerbarhet
- Utvidbarhet

Ved å forenkle der det er mulig ønsker vi å kunne forbeholde komplekse løsninger der de virkelig trengs. En enkel løsning vil også ha færre meldinger som utveksles mellom nodene, og dette kan derfor påvirke skalerbarheten. I tillegg innebærer enkelhet at antall tilstander og antall overganger mellom tilstander ikke bør overdrives. Enkelhet bidrar til at gjenoppretting etter feil kan forenkles, noe som igjen medfører at gjenopprettingen kan skje raskt.

Siden vi antar at omgivelsene som metodene skal benyttes i er upålitelige, må vi også anta at feilsituasjoner oppstår ofte. Et viktig element ved multinodestreaming er at datatilgjengeligheten skal opprettholdes eller hurtig gjenopprettes etter at feilsituasjoner har oppstått. Vi anser derfor feilhåndtering som en kjernefunksjonalitet, og vi ønsker derfor å ta hensyn til dette for å lage

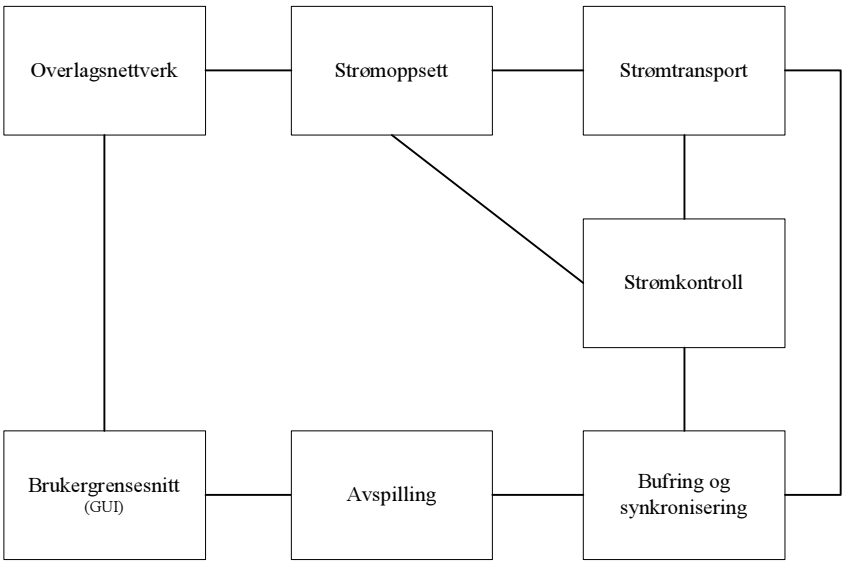
systemet mest mulig feiltolerant. I tillegg må systemet håndtere feilsituasjoner mest mulig effektivt.

Skalerbarhet er et viktig designprinsipp for alle P2P-systemer, og dette er også noe vi ønsker å ta hensyn til i metodespesifikasjonen. Et skalerbart P2P-system med et stort antall noder vil forhåpentligvis også inneholde et stort antall replikater av hvert objekt, noe som igjen vil påvirke tilgjengeligheten positivt.

For å gjøre metodene utvidbare for et vidt applikasjonsspekter, ønsker vi å utsette avgjørelser som ikke er en del av metodenes hovedfokus. Ved ikke å bestemme om metodene avhenger av et sentralisert eller desentralisert overlagsnettverk holdes mulighetene åpne for begge, og dermed utsettes også til dels avgjørelsen om grad av autonomi. Utsatte avgjørelser brukes derfor for å tilfredsstille designprinsippet om utvidbarhet.

5.2.3 Overordnet design

Vårt system for multinodestreaming vil bestå av et rammeverk av moduler som organiseres som vist i figur 5.4. Enkelte av modulene vil i all hovedsak baseres på standardkomponenter som er tilgjengelige i programvarebiblioteker, mens andre komponenter må implementeres eksplisitt for vår metode.



Figur 5.4: Overordnet arkitektur.

Overlagsnettverk Et generelt grensesnitt mot alle typer overlagsnettverk tilbys av modulen Overlagsnettverk, og denne modulen må i tillegg også inne-

holde en implementasjon av et overlagsnettverk som kan utføre metodene som er spesifisert i grensesnittet. Hensikten med et generelt grensesnitt er å gjøre systemet uavhengig av et spesifikt overlagsnettverk samt å forenkle eksperimentering med ulike typer overlagsnettverk. Overlagsnettverket brukes for å lokalisere ressurser, og dette gjøres ved at en forespørsel sendes ut og rutes via andre noder fram til mottakernoden.

Strømoppsett Før en avspillingssesjon startes, konfigureres transport av mediestrømmer av modulen Strømoppsett. Modulen er ansvarlig for å velge hvilke noder mediestrømmer skal hentes fra. For å få bekreftet at en avspillingssesjon kan etableres, må denne modulen også kunne forhandle med de potensielle leverandørene av hver mediestrøm. Denne modulen oppretter også en liste over reservenoder som kan overta dersom en node som leverer en mediestrøm skulle feile.

Strømtransport Rutiner for streaming av kontinuerlige medier finnes i Strømtransport, og denne modulen er derfor ansvarlig for å transportere mediedata, samt avlevere rapporter om leveransen eller mottaket sin status. Slik funksjonalitet finnes allerede i eksisterende Internett-protokoller, og denne modulen kan sannsynligvis baseres på ferdigkomponenter.

Strømkontroll Overvåkning av strømmottak utføres av Strømkontroll, og dersom en avsender blir utilgjengelig eller rapporten fra strømtransportmodulen tilsier at leveransen bør avbrytes, sørger denne modulen for at leveransen av den avbrutte strømmen blir overført til en av reservenodene.

Bufring og synkronisering Multipleksing av innkommende delstrømmer slik at de samles i én buffer utføres av Bufring og synkronisering. Den samlede bufferen inneholder mediedataene som brukes av avspillingsmodulen.

Avspilling Dekoding av den samlede mediestrømmen, slik at strømmen kan presenteres for brukeren, foregår i Avspilling. Denne modulen består hovedsakelig av komponenter som kan dekode de aktuelle medieformatene som systemet streamer, samt moduler som kan presentere medieformater for sluttbrukere.

Brukergrensesnitt Presentasjon av selve avspillingen utføres av modulen Brukergrensesnitt, og modulen tilbyr i tillegg brukeren å søke etter og iverksette avspilling av medieobjekter. Denne modulen er dermed opphavet til samtlige brukerinitialiserte operasjoner.

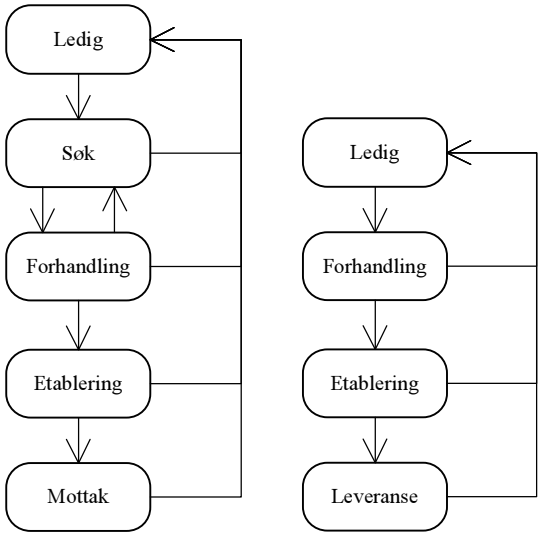
5.3 Metodespesifikasjon

Vi har nå laget et rammeverk for en multinodestreamingbasert metode ved å spesifisere funksjonelle egenskaper og overordnet design. De funksjonelle egenskapene legger grunnlaget for hvilke funksjoner et komplett system må implementere. Ut fra det overordnet designet vil vi videre spesifisere konkrete tilstander, meldinger og prosesser. Denne spesifikasjonen danner grunnlaget for en implementasjon, samt analytiske vurderinger av metodens funksjon og verdi.

I den videre spesifikasjonen vil vi legge vekt på de modulene som inngår i nodeadministrasjonslaget. Eksterne komponenter (omgivelsene), som for eksempel overlagsnettverk, er ikke en del av denne spesifikasjonen. Målet er å spesifisere en metode som bidrar til å maksimere opplevd datatilgjengelighet, og som samtidig tar hensyn til punktene angående designfilosofi beskrevet i kapittel 5.2.2. Av de funksjonelle egenskapene beskrevet i tabell 5.2 er det i hovedsak punktene 3 ("Systemet må kunne å motta flere mediestrømmer samtidig"), 4 ("Systemet må kunne synkronisere ulike mediestrømmer i forhold til hverandre") og 5 ("Endringer i omgivelsene som påvirker ressurstilgjengelighet skal føre til tiltak som kan opprettholde eller gjenopprette tilgjengeligheten hvis dette er mulig") som er sentrale i forhold til nodeadministrasjon. For punkt 3 og 4 er det kun funksjonaliteten rundt det å administrere forskjellige noder som er sentrale, ikke funksjonaliteten knyttet til streaming eller annen nettverkskommunikasjon.

Overordnede tilstander

Et system vil bestå av både mottaker- og avsendernoder. Hver node kan befinne seg i forskjellige tilstander som både mottaker og avsender avhengig av hvilke oppgaver de utfører. Figur 5.5 viser overgangene mellom tilstandene som systemet befinner seg i når det opptrer som mottaker eller avsender. Felles for både mottaker og avsender er at alle tilstander etter feilsituasjoner har en mulig overgang til startilstanden (Ledig).



(a) Mottaker (b) Avsender

Figur 5.5: Overordnet tilstandsdiagram.

Mottaker En node som ønsker å motta et medieobjekt starter med å utføre et søk etter noder som tilbyr det aktuelle medieobjektet som ønskes avspilt ("Søk"). Deretter sender mottaker ut forespørsler om leveranse av delstrømmer til potensielle avsendernoder ("Forhandling"). Avsendernodene svarer ved å bekrefte eller avkrefte at de er i stand til å levere de forespurte delstrømmene. Når mottaker har fått bekreftelse om at samtlige delstrømmer kan leveres, går den over til neste fase ("Etablering"). Alternativt må mottaker utføre et nytt søk hvis den ikke finner et tilstrekkelig antall noder som kan levere de ønskede delstrømmene. Mottaker etablerer deretter, hos hver avsender, tilkoblinger for transportprotokollene som skal transportere delstrømmene. Når alle delstrømmer er etablert sender mottaker startsignal til avsenderne, og leveransen starter ("Mottak").

Avsender En avsendernode er passiv inntil en den får en forespørsel fra mottaker ("Forhandling"). Avsender avgjør da om den har nok ledig kapasitet til å levere den forespurte delstrømmen, og sender da enten bekreftelse eller avkreftelse om at mediestrømmen kan leveres. Videre avventer avsender at mottaker skal etablere en tilkobling med transportprotokollene ("Etablering"), og deretter at mottaker skal gi signal om at leveransen skal starte ("Leveranse").

Antagelser og forenklinger

Vi antar at alle noder har lik sannsynlighet for å bli utilgjengelig, og derfor vil metoden for forhandling forsøke å hente hver delstrøm fra ulike noder. Når en node forsvinner vil den derfor kun påvirke én av delstrømmene. Alternativt kan noder som har tilstrekkelig ledig båndbredde levere flere delstrømmer. For mottaker innebærer dette imidlertid en ulempe med hensyn på ressurstilgjengelighet. For at dette skal innføres bør mottaker derfor ha tilgang til informasjon om andre noder sin nodetilgjengelighet, og det er noe som foreløpig vil gjøre forhandlingen unødvendig kompleks.

Siden alle noder kan opptre som både mottakere og avsendere, antar vi at både avsender- og mottakernoder har lik sannsynlighet for å bli utilgjengelige. Dette innebærer at en avsender sine løfter til en mottaker kun gjelder for visse perioder av gangen, og for å opprettholde tilstander hos en avsender må en mottaker oppdatere disse med jevne mellomrom. Dersom en mottakernode går ned blir ressursene som den har reservert hos avsendere frigjort etter maksimalt én oppdateringsperiode.

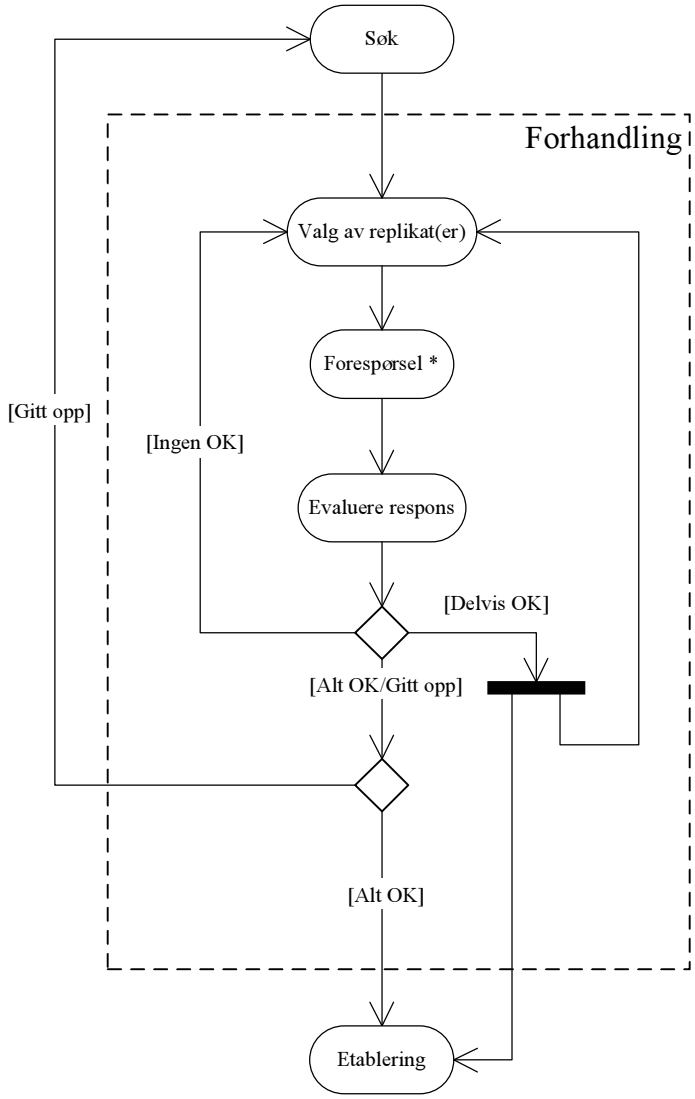
Fordi multinodestreaming i all hovedsak styres av mottakernoden vil vi konsentrere oss om å spesifisere funksjonalitet på mottakersiden. Funksjonaliteten som trengs for at en node skal være i stand til å levere en delstrøm skiller seg i liten grad fra hva som kreves for ordinær streaming, og vil derfor ikke bli videre omtalt i spesifikasjonen.

5.3.1 Forhandling

Etter at mottaker har utført et søk, og dermed funnet noder som tilbyr et ønsket medieobjekt, må mottaker forhandle med potensielle avsendere. Hensikten med forhandlingen er at mottaker, før leveransen starter, skal undersøke om nodene som tilbyr medieobjektet faktisk er i stand til å levere det. Forhandlingen skal med andre ord bidra til å unngå situasjoner der mottaker forsøker å hente en delstrøm fra en avsendernode som ikke er i stand til å levere den.

Forhandlingsprosessen er illustrert i figur 5.6. Forhandlingen starter med utgangspunkt i resultatet av et søk, og dette resultatet er en liste over noder som tilbyr medieobjektet. I aktiviteten “Valg av replikat(er)” (dekomponert i figur 5.7(b)) vil mottaker sortere denne listen etter hver node sin antatte grad av leveransedyktighet (“Sortere replikatliste”). Noder som allerede leverer en delstrøm til mottaker blir fjernet fra listen (“Fjerne replikater i bruk”). Sorteringskriteriet for listen kan for eksempel være basert på rundetid (*eng. “Round-Trip-Time”, RTT*) eller antall ruterhopp mellom mottaker og avsender. Sorteringskriteriet brukes dermed som en heuristikk i forhandlingen, og målet er at mottaker så tidlig som mulig skal starte forhandlinger med de

nodene som faktisk er de best egnede avsenderne ("Velge beste").

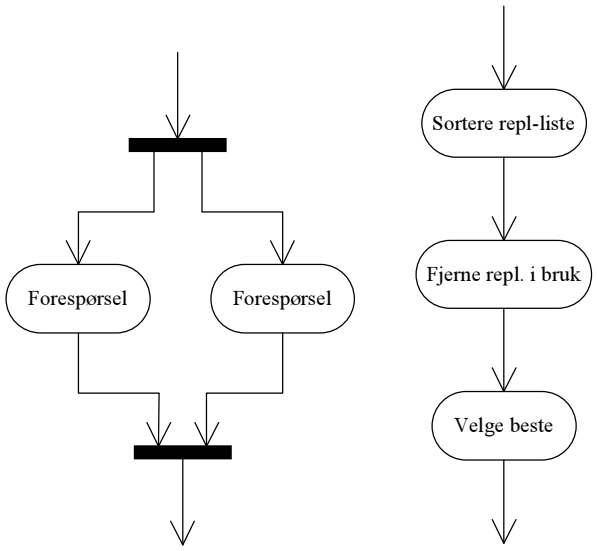


Figur 5.6: Forhandlingsprosessen (UML aktivitetsdiagram).

Etter at valg av replikater er utført sender mottaker, samtidig for hver delstrøm, forespørsler til mulige avsendernoder ("Forespørsel", dekomponert i figur 5.7(a)). At denne aktiviteten utføres i parallell for flere noder vises i figur 5.6 med tegnet "*" etter aktivitetsnavnet. Neste aktivitet er "Evaluere respons", og dersom ingen av de forespurte avsenderne er i stand til å levere må "Valg av replikater" utføres på nytt. Dersom kun et utvalg av de forespurte nodene er i stand til å levere fortsetter disse til neste aktivitet ("Etablering"). Samtidig forsøkes de delstrømmene som fortsatt ikke har noen bekreftet avsender å bli reforhandlet ("Valg av replikater"). Reforhandlingen fortsetter inntil den ikke

kan fortsette med den gjeldende replikatlisten, og dermed må "Søk" utføres på nytt.

I tilfellet der alle delstrømmene kan leveres fra de forespurte nodene fortsetter alle disse til "Etablering", og forhandlingsfasen avsluttes.

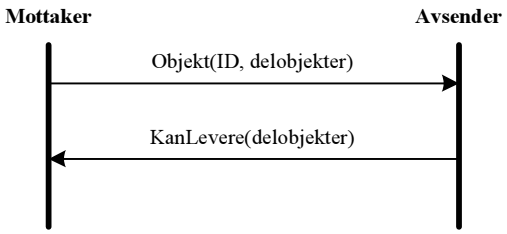


(a) Forespørsel.

(b) Valg av replikater.

Figur 5.7: Dekomponerte aktiviteter (UML aktivitetsdiagram).

Meldingene som utveksles mellom avsender og mottaker i aktiviteten "Forespørsel" er vist i figur 5.8. Mottaker forespør avsender om den er i stand til å levere et eller flere delobjekter (eller delstrømmer) av et objekt med en oppgitt identifikator. Avsender svarer med å oppgi hvilke delstrømmer som kan leveres. Dersom delobjekter kan være av ulik størrelse kan det også tenkes at mottaker forespør et delobjekt som er for stort til at avsender kan levere det, men i stedet for å avvise det forespurte delobjektet kan avsendernoden i stedet oppgi eventuelle andre delobjekter som den er i stand til å levere.



Figur 5.8: Meldinger ved utsendelse av forespørsel.

For å gjøre beskrivelsen av forhandlingen mer presis, og for å beskrive metoden på et nivå som kan benyttes i en implementasjon, har vi også beskrevet den i pseudokode i algoritme 5.1.

funksjon Forhandle()
Startbetingelse: Søk() har resultert i minst ett treff
1: **hvis** maksimalt antall forhandlingsforsøk **så**
2: Søk() #gjeldende forhandling avsluttes
3: **slutt hvis**
4: ranger avsendernoder
5: **for alle** uetablerte delstrømmer **utfør**
6: tilordne beste ledige avsendernode
7: **slutt for**
8: **for alle** noder tilordnet delstrøm **utfør**
9: send forespørsel
10: motta respons
11: **slutt for**
12: **for alle** OK delstrømmer **utfør**
13: Etabler()
14: **slutt for**
15: **for alle** ikke OK delstrømmer **utfør**
16: fjern tilordnet avsendernode
17: Forhandle() #fortsetter forhandling
18: **slutt for**

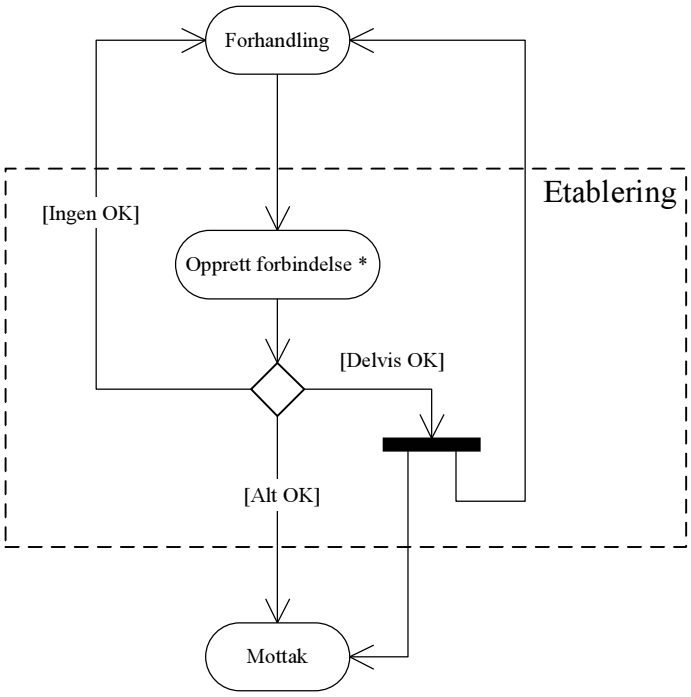
Algoritme 5.1: Pseudokode for forhandlingsfasen.

5.3.2 Etablering

I forhandlingsfasen etablerer mottakernoden en forhandlingsprosess med potensielle avsendernoder. Hvis forhandlingene lykkes, det vil si at avsendernoden er i stand til og ønsker å sende ett eller flere delobjekter til mottakernoden, vil mottakernoden kunne gå videre til etableringsfasen for å opprette en le-

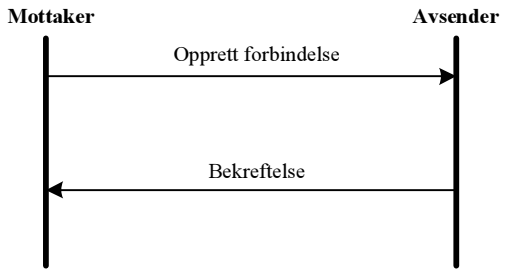
veransesesjon. Etableringsfasen tar til når mottakernoden har lykkes med å forhandle fram leveranse av et tilstrekkelig antall delobjekter til at etablering og mottak (og dermed også avspilling) kan begynne. Dette kan skje til tross for at det ikke har lykkes å fremforhandle leveranse av samtlige delobjekter. Mottaker kan selv avgjøre hvor stor andel av delobjektene som må være klar til levering for at etableringsfasen kan ta til. Mottakernoden kan fortsette å søke etter replikater av de resterende delobjektene (og forhandle om leveranse av disse) samtidig som den etablerer leveransesesjoner for de øvrige delobjektene. Dermed kan kvaliteten på avspillingen økes etter hvert, og avspilling hos mottakeren kan tiltre tidligere enn om den skulle vente på alle delobjektene.

Etablering av en sesjon vil si at det utveksles sesjonsdata og opprettes forbindelser for de underliggende transportprotokollene (for eksempel RTP). Figur 5.9 illustrerer denne prosessen. Etter forhandling har lykkes, starter mottakernoden prosessen med å opprette forbindelser til hver avsendernode. Etter at det har blitt gjort forsøk på å etablere forbindelser med alle nodene (som forhandlingene lykkes med), evalueres resultatet. I noen tilfeller kan det ha mislykkes i å opprette en forbindelse (for eksempel hvis en node har meldt seg ut av nettverket). I slike tilfeller må det innledes nye forhandlinger om de aktuelle delobjektene. Forbindelsene som det lykkes å etablere vil umiddelbart bli brukt til strømmottak så lenge det er et tilstrekkelig antall forbindelser til at avspilling kan utføres med tilfredsstillende kvalitet. Dette kan skje samtidig med at det innledes forhandlinger om delobjekter som likevel ikke kunne leveres. Systemet må til enhver tid vurdere om det kan leveres et tilstrekkelig antall delobjekter. Hvis dette antallet blir for lavt, kan mottakernoden velge å avbryte leveransen i stedet for å starte avspillingen med kun en delmengde av delobjektene.



Figur 5.9: Etableringsprosessen (UML aktivitetsdiagram).

I etableringsfasen vil det utveksles meldinger over nettverket mellom mottaker- og avsendernodene. Disse meldingene brukes for å sette opp forbindelsene i transportlaget. Figur 5.10 illustrerer meldingsaktivitetene i denne fasen. Detaljer orientert rundt spesifikke protokoller er ikke tatt med her, blant annet for å ikke legge direkte føringer på valg av transportprotokoll. I den videre diskusjonen antar vi imidlertid at RTP brukes som transportprotokoll, blant annet fordi RTP er mest egnet til å sette opp avanserte streamingsesjoner, og sammen med protokollen RTCP gir den mulighet til å kontrollere leveransen av en delstrøm. Slik kontroll kan blant annet bli nødvendig i forbindelse med synkronisering. I figuren initialiserer mottakernoden opprettelsen av forbindelsen, og avsendernoden svarer med å sende en bekreftelse. På dette stadiet er en forbindelse opprettet uten at den brukes for å streame data over den. Selve dataleveransen gjøres først etter at mottakernoden har sendt en ny melding i neste fase: Mottak.



Figur 5.10: Meldinger mellom mottaker og avsender under etablering.

Unntakene (feilene) som kan oppstå i etableringsfasen medfører at nye forhandlinger må innledes. Årsakene til at et unntak oppstår kan være hva som helst (nettverksfeil, node melder seg ut osv.). Skjer det feil etter at etableringen har lykkes, er det opp til unntakshåndteringen i mottaksfasen å iverksette tiltak for å gjenopprette leveranse av de berørte delobjektene. Når feilene skjer etter at forhandlinger er gjennomført og avtaler om leveranse er inngått, men likevel før mottak, blir alltid resultatet at det innledes nye forhandlinger om de aktuelle delobjektene. Etableringsfasen er også beskrevet i algoritme 5.2 som pseudokode.

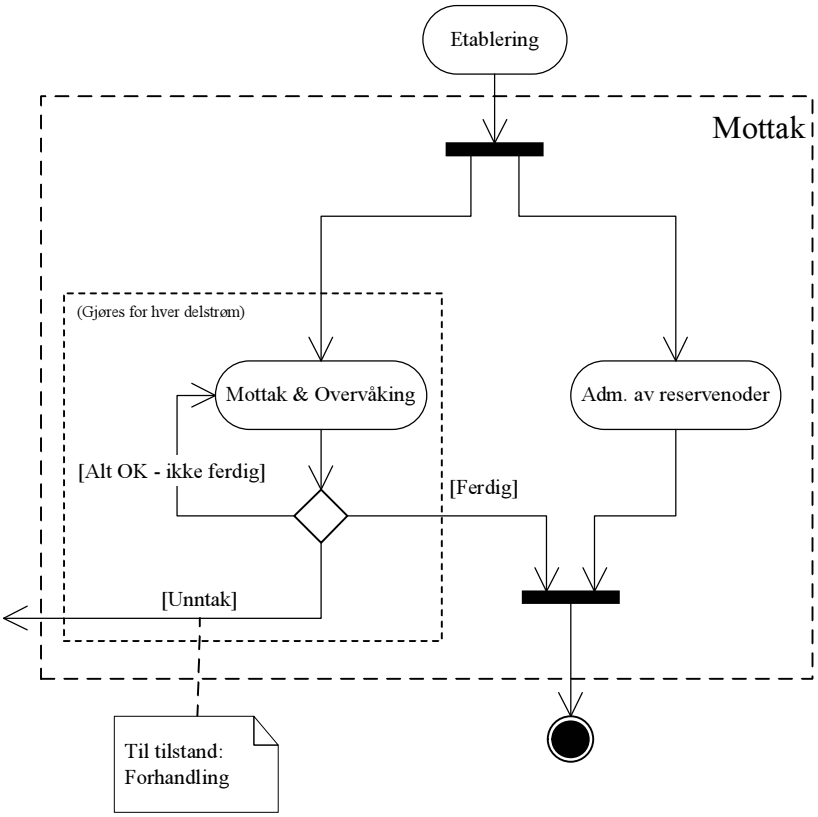
```
funksjon Etabler()
Startbetingelse: Forhandle() var vellykket for minst én node.
 1: for alle avsendernoder utfør
 2:    opprett forbindelse
 3:    evaluer resultat    #ble forbindelse opprettet?
 4: slutt for
 5: hvis ingen forbindelser OK så
 6:    for alle delobjekter utfør
 7:       Forhandle()
 8:    slutt for
 9: eller hvis noen forbindelser OK så
10:    for alle OK forbindelser utfør
11:       Motta()
12:    slutt for
13:    for alle ikke OK delobjekter utfør
14:       Forhandle()
15:    slutt for
16: ellers
17:    for alle forbindelser utfør
18:       Motta()
19:    slutt for
20: slutt hvis
```

Algoritme 5.2: Pseudokode for etableringsfasen.

5.3.3 Mottak

Mottak startes etter at en delstrøm har blitt etablert, og dens oppgave er å starte og opprettholde mottak av delstrømmer. Samtidig som delstrømmene mottas vil mottaksprosessen også overvåke strømmene. For at systemet skal være forberedt på eventuelle feilsituasjoner der leveransen må overføres til nye avsendernoder, vedlikeholder mottaksprosessen også en liste over reservenoder som har mulighet til å levere medieobjektet som mottas.

Figur 5.11 illustrerer mottaksprosessen. To aktiviteter, “Mottak & Overvåking” og ”Adm. av reservenoder”, utføres i parallell. Når mottaksprosessen er ferdig, avsluttes alle aktiviteter i systemet (symbolisert med en sluttilstand).



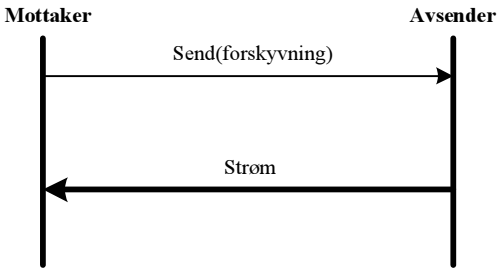
Figur 5.11: Mottaksprosessen (UML aktivitetsdiagram).

Mottak & overvåkning

Aktiviteten "Mottak & overvåkning" utføres for hver delstrøm, og når aktiviteten startes må også selve dataleveransen starte. Oppstart utføres ved at avsender starter overføringen av data via tilkoblingen som ble opprettet under etableringsfasen. Både mottak og overvåkning utføres av eksisterende transport- og kontrollprotokoller for sanntidsmedier. På grunnlag av overvåkningsrapporten fra kontrollprotokollen må systemet avgjøre om avsendernode ikke lenger er i stand til å levere. Dermed avsluttes mottaket av den feilede delstrømmen samtidig som feilhåndtering starter. Dersom overvåkningsrapporten ikke tilsier at noe er galt, fortsetter mottaket. Når delstrømmen er ferdig avspilt avsluttes mottak og overvåkning. Dersom alle andre delstrømmer av samme medieobjekt også er ferdig avspilt vil også prosessen for administrasjon av reservenoder avsluttes.

Meldingene som utveksles vises i figur 5.12. Dersom delstrømmen tilhører et medieobjekt som allerede avspilles, må avspillingen av den nye delstrømmen starte fra samme posisjon som de andre delstrømmene befinner seg i, og det er

årsaken til forskyvningsparameteren til send-meldingen.



Figur 5.12: Meldinger mellom mottaker og avsender under mottaksprosessen.

Administrasjon av reservenoder

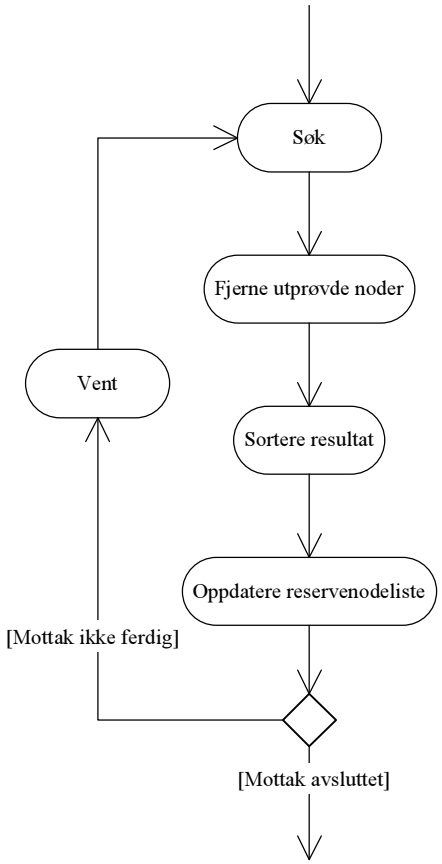
Ved mottak vil et objekt leveres av en eller flere noder i nettverket. Hvis en av disse nodene blir utilgjengelig, vil en reservenode brukes i stedet. Administrasjon av reservenoder er en prosess som kontinuerlig vedlikeholder en oppdatert oversikt over andre noder i nettverket som kan levere det aktuelle objektet. Hvis en en delstrøm blir utilgjengelig, vil reservenodeadministrasjonen allerede ha bestemt hvilke noder som skal forespørres om å overta ansvaret for leveransen. Målet med denne prosessen er å gjøre gapet mellom faktisk- og opplevd datatilgjengelighet så liten som mulig. Jo lenger tid det tar å gjøre overtagelsen for leveranse av en delstrøm, desto større blir gapet mellom faktisk- og opplevd datatilgjengelighet. Reservenodeadministrasjonen tar derfor sikte på å effektivisere feilhåndtering, og på den måten gjøre den opplevde tjenestekvaliteten (med tanke på tilgjengelighet) bedre.

Administrasjonen gjøres for samtlige delstrømmer, og er derfor en fellestjeneste. Feilhåndtering er derimot en prosess som gjøres per delstrøm. Prosessen som administrerer reservenoder benytter en reservenodeliste. Reservenodelisten oppdateres periodisk, og inneholder til enhver tid en sortert oversikt over reservenoder. Sorteringen gjøres på basis av de samme kriteriene som replikatlisten beskrevet i kapittel 5.3.1. I tillegg fjernes noder som allerede har vært utprøvd som reservenoder.

Et alternativ til å administrere en reservenodeliste basert på resultater fra overlagsnettverket er å på forhånd forhandle fram reservenodeavtaler med aktuelle noder. En slik løsning ville i større grad sikret at en reservenode var i stand til å levere et objekt, men ville også reservert uforholdsmessig mye ressurser i nettverket. En slik løsning hemmer skalerbarheten til systemet, og strider derfor med et av våre opprinnelige designmål.

Prosessen ved å administrere reservenoder er illustrert i figur 5.13. Så lenge mottaket av et objekt pågår, fortsetter administrasjonen av reservernoder.

Dette er altså en kontinuerlig prosess, illustrert ved at prosessen gjenopptas (etter en ventetid) etter at hver runde er fullført.



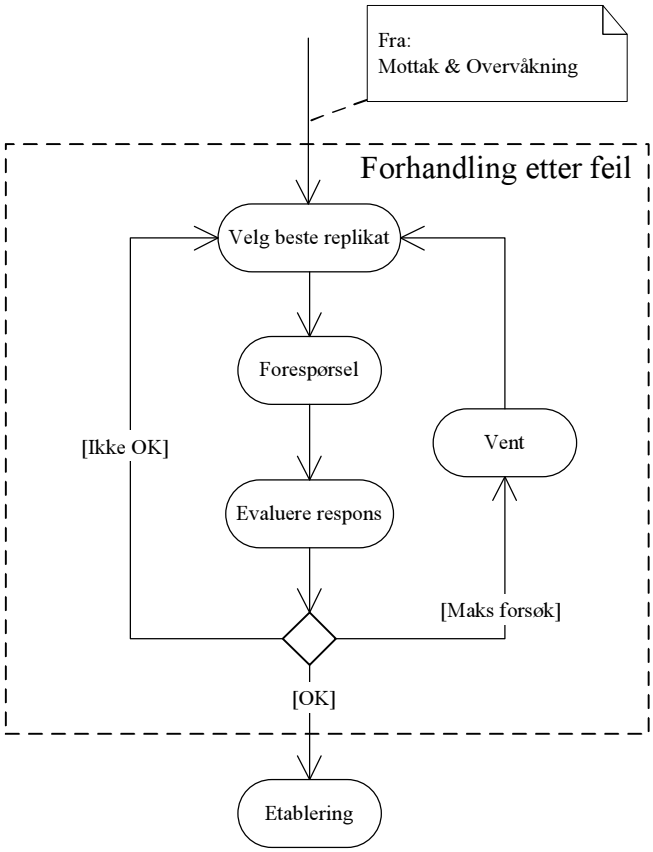
Figur 5.13: Dekomponering av aktiviteten "Adm av reservenoder" (UML aktivitetsdiagram).

Den første prosessen under administrasjonen av reservenoder er "Søk". Søket gjøres mot overlagsnettverket, og resultatet behandles ved at utprøvde noder fjernes og de resterende nodene sorteres ut fra bestemte kriterier. Resultatet legges i reservenodelisten som deretter kan brukes ved feilhåndtering. En ventetid legges inn før neste runde med søk og prosessering. Dette gjøres for å ikke belaste overlagsnettverket unødig. Hvis nodetilgjengelighetssituasjonen ikke endres veldig ofte, vil det heller ikke være nødvendig å gjøre etterfølgende søk for å ha en tilnærmet oppdatert reservenodeliste.

5.3.4 Feilhåndtering

Feilhåndtering er prosessen som gjenoppretter leveranse av en delstrøm etter at en feil er oppdaget. Leveransen feiler når en node som leverer en delstrøm ikke lenger er i stand til å levere, og dette kan for eksempel skyldes at noden melder seg ut av systemet eller at nettverket blir mettet.

Prosessen for feilhåndtering utføres per delstrøm som har feilet, og er illustrert i figur 5.14. Feilhåndtering er egentlig en variant av prosessen "Forhandling", men siden feilhåndteringen må utføres i løpet av en kort tidsperiode, vedlikeholdes listen over reservenoder jevnlig av en annen prosess som administrerer reservenoder. Feilhåndteringen kan dermed starte direkte ved å velge den antatt best egnede reservenoden ("Velg beste replikat") for deretter å sende en forespørsel til denne ("Forespørsel"). Deretter evalueres responsen ("Evaluere respons"). Feilhåndtering utføres per delstrøm, og ikke per medieobjekt slik som forhandlingsfasen. Utfallet av evalueringen blir derfor enten at delstrømmen kan leveres eller at neste reservenode må forespørres. Dersom feilhåndteringen har blitt forsøkt utført et visst antall ganger uten suksess, eller hvis alle reservenodene har sendt negativ respons, vil systemet gi opp og vente på at eventuelle nye reservenoder skal bli identifisert av prosessen for administrasjon av reservenoder (i aktiviteten "Vent").



Figur 5.14: Forhandlingsprosessen ved feilhåndtering (UML aktivitetsdiagram).

Meldingene som utveksles mellom nodene ved feilhåndtering er identiske med meldingene som sendes mellom nodene under forhandling (figur 5.8), og de blir derfor ikke videre beskrevet her. Pseudokode som beskriver feilhåndteringen er gitt i algoritme 5.3.

```
funksjon Feilhåndter()
Startbetingelse: En leveransefeil har blitt oppdaget
 1: så lenge uprøvde reservenoder eksisterer utfør
 2:    velg beste ledige avsendernode
 3:    send forespørsel
 4:    motta respons
 5:    hvis respons OK så
 6:       Etabler()    #feilhåndtering avsluttes
 7:    eller hvis respons ikke OK så
 8:       merk avsendernode som utprøvd
 9:       hvis maksimalt antall forsøk så
10:          vent    #venter på oppdatert reservernodeliste
11:       slutt hvis
12:       Feilhåndter()    #gjentar feilhåndtering
13:    slutt hvis
14: slutt så lenge
15: vent    #venter på oppdatert reservernodeliste
16: Feilhåndter()    #gjentar feilhåndtering
```

Algoritme 5.3: Pseudokode for feilhåndtering.

Av pseudokoden i algoritme 5.3 går det fram at feilhåndteringen ikke avsluttes før en reservenode som er i stand til å levere den ønskede delstrømmen er blitt funnet (linje 6). Dersom feilhåndteringen ikke lykkes skyldes dette enten at antall forsøk på å finne en reservenode er overskredet (linje 10) eller at ingen reservenoder er i stand til å levere den forespurte delstrømmen (linje 15). I begge tilfeller vil prosessen for feilhåndtering stoppe opp for å vente på at prosessen for administrasjon av reservenoder skal oppdatere listen over reservenoder. Venteperioden må være minimum lik perioden som prosessen for administrasjon for reservernoder opererer med, men den kan også settes slik at den øker med antall forsøk på feilhåndtering. En mottaker som ikke klarer å gjenopprette en feil vil dermed ikke belaste andre noder unødvendig.

5.4 Evaluering

Et av de potensielle problemene knyttet til streaming av digitale medier i P2P-nettverk er usikker datatilgjengelighet. Hvis en node blir utilgjengelig samtidig som den er leverandør av et objekt til en annen node, vil den mottakende noden oppleve et leveransebrudd. For kontinuerlige medier som video og lyd er et slikt avbrudd kritisk for den opplevde tjenestekvaliteten. I dette kapittelet har vi spesifisert en nodeadministrasjonsmetode som har som mål å minimere gapet mellom opplevd og faktisk datatilgjengelighet i et P2P-nettverk. Denne

metoden tar utgangspunkt at det i mange P2P-nettverk vil være stor grad av objektreplikering. Det er derfor stor sannsynlighet for at flere enn én node er i stand til å levere et bestemt objekt. Løsningen vi har spesifisert i dette kapittelet er todelt:

1. Ved å bruke "multinodestreaming" vil ulike deler av et objekt leveres av forskjellige noder. Kvaliteten på avspillingen bestemmes av antallet delobjekter som mottas. Ett enkelt strømbrudd påvirker altså bare avspillingskvaliteten og ikke hele avspillingen.

2. Systemet er i stand til å reagere raskt ved strømbrudd slik at full avspillingskvalitet gjenopprettes i løpet av kort tid (så lenge den faktiske ressurstilgjengeligheten i nettverket tillater det). Effektiv unntakshåndtering gjøres ved at mottakernoden utfører en kontinuerlig overvåking av innkommende strømmer. Samtidig administreres en liste over reservenoder som hurtig kan overta hvis en strøm enten har for dårlig kvalitet (for eksempel ved stort pakketap) eller stanser fullstendig.

5.4.1 Evalueringskriterier

I dette delkapittelet tar vi sikte på å gjøre en helhetlig vurdering av den foreslåtte metoden med det mål å vurdere om den er egnet ut fra tidligere spesifisert designfilosofi, krav til funksjonalitet og effekt i forhold til målet om å øke den opplevde datatilgjengeligheten. Dette gjør vi ved å først fastslå hvilke evalueringskriterier som skal legges til grunn for denne vurderingen. Deretter vil metoden vurderes opp mot de fastsatte kriteriene.

Evalueringen av metoden skal legge følgende faktorer til grunn:

1. Samsvar med generelle P2P-prinsipper. Det er viktig at metoden ikke bryter med viktige karakteristikker ved P2P-arkitekturen, blant annet autonomi og direkte kommunikasjon mellom noder. Generelle P2P-prinsipper er blant annet beskrevet i kapittel 2.1.

2. Hvorvidt metodespesifikasjonen er i samsvar med tidligere fastsatt designfilosofi.

3. Hvilke funksjonelle egenskaper som metoden implementerer i forhold til dem som ble fastsatt i tabell 5.2.

4. Hvordan metoden påvirkes av omgivelsene. Dette inkluderer blant annet å vurdere hvilke eksterne funksjoner metoden er avhengig av, og hvor kritiske disse er (funksjon, effektivitet osv.) for at metoden skal fungere tilfredsstillende.

5. Hvorvidt det er mulig å implementere metoden i et ekte P2P-system. Er løsningen implementerbar, og vil den fungere i eksisterende nettverks- og maskinvareomgivelser?

6. Gitt at metoden er implementerbar (det vil si at den funksjonelt sett er mulig å realisere): Hvor effektiv er metoden med tanke på å gjenopprette maksimal kvalitet (gitt at den faktiske ressurstilgjengeligheten i nettverket er tilstrekkelig for å utføre en slik gjenopprettelse). Hvor lang tid bruker hver fase i gjenopprettingen fra en feil er oppdaget til normaltilstand er gjenopprettet?

Ved selve evalueringen vil hvert evalueringskriterium kunne bli utdypet hvis det er nødvendig. Enkelte evalueringskriterier vil være vanskelige å vurdere uten å ta en bestemt kontekst eller omgivelsesspesifikasjon i betraktning. I slike tilfeller vil vi måtte gjøre antagelser, noe som vil bli spesifisert der det er aktuelt.

5.4.2 Evaluering av metoden

Evalueringskriteriene gitt i forrige avsnitt vil for kriteriene 1 – 4 kunne vurderes opp mot metodespesifikasjonen beskrevet tidligere i dette kapittelet. Noen evalueringskriterier krever imidlertid en implementert løsning for å kunne vurderes ordentlig. Dette gjelder kriteriene 5 og 6. Disse kriteriene vil vi komme tilbake til i kapittel 6 som blant annet beskriver en implementert prototyp hvor vi tester de forskjellige konseptene knyttet til multinodestreaming og unntakshåndtering.

Samsvar med generelle P2P-prinsipper

Den spesifiserte metoden tar utgangspunkt i problemene knyttet til leveranse av kontinuerlige medier i tradisjonelle P2P-systemer, og tar dermed også utgangspunkt i en arkitektur som antas å være i samsvar med generelle P2P-prinsipper. Multinodestreaming er ett av to funksjonelle tillegg som metoden introduserer i tillegg til eksisterende P2P-funksjonalitet. I tillegg introduseres funksjonalitet for kontinuerlig administrasjon av reservenoder og unntakshåndtering. I metoden har det vært lagt vekt på at det er mottakernoden som organiserer og koordinerer dataleveranse under normaltilstand (når det ikke er en unntakssituasjon under behandling). Også administrasjon av reservernoder og håndtering unntakssituasjoner er mottakernodens ansvar. Metoden er altså i fullt samsvar med P2P-prinsippet om autonomi (selvstyre), det vil si at kontroll ikke overlates til sentraliserte eller andre utenforstående enheter, både under normal- og unntakstilstander. De øvrige P2P-prinsippene berøres ikke direkte av metodespesifikasjonen.

Samsvar med designfilosofi

I kapittel 5.2.2 beskrev vi fire hovedpunkter som var viktige for vår designfilosofi. Disse hovedpunktene var:

- Enkelhet
- Feiltoleranse
- Skalerbarhet
- Utvidbarhet

Den spesifiserte metoden beskriver kun funksjonaliteten som faller inn under nodeadministrasjonslaget, men må som delsystem også følge den samme filosofien som et totalsystemet (som også inkluderer de øvrige komponentene i P2P-systemet).

Et av de viktigste designprinsippene er enkelhet. I metodespesifikasjonen ble det lagt spesiell vekt på å ha få tilstander, og at det skulle være enkle (få) overganger mellom tilstandene. Mottakerdelen av systemet utfører selve nodeadministrasjonen, og er derfor viktigst i forhold til evalueringen av metoden. For mottakerdelen er hovedtilstandene: Ledig, Søk, Forhandling, Etablering, og Mottak. Disse tilstandene er tilstrekkelige for å beskrive systemets virkemåte både under normal kjøring og ved håndtering av feil. Det er også få nettverksmeldinger for hver fase, og enkel flytlogikk som beskriver de ulike prosessene i systemet. Dette er i samsvar med designprinsippet om enkelhet.

Feiltoleranse er ivaretatt ved at systemet har funksjoner som gjenoppretter normaltilstand etter at et unntak har funnet sted uten at det påvirker tilstandene knyttet til de øvrige delstrømmene i systemet. En feil påvirker altså en så liten del av systemet som mulig samtidig som systemet er i stand til å effektivt gjenopprette normaltilstand (dersom omgivelsene tillater det). Et fullstendig P2P-system sett under ett er også feiltolerant i den forstand at det ikke er noen noder som alene kan bringe systemet ned. Dette er imidlertid en egenskap som er knyttet til P2P-arkitekturen generelt, og er ikke en spesiell kvalitet knyttet til vår metodespesifikasjon.

Skalerbarhet er en av de generelle styrkene til P2P-systemer sammenlignet med tradisjonelle klient/tjener-arkitekturer. Årsaken til at P2P-systemer skalerer godt er at de ikke er avhengig av sentraliserte enheter. I tillegg har P2P-systemer en naturlig likevekt mellom ressurser som tilbys til resten av nettverket og ressurser som konsumeres (fordi en node bidrar med ressurser i tillegg til å konsumere dem). I spesifikasjonen av nodeadministrasjonsmetoden har det blitt lagt vekt på at en node ikke skal være avhengig av sentraliserte eller andre eksterne enheter for å fungere. Metoden legger altså ingen åpenbare begrensninger i forhold til å nå målet om skalerbarhet.

Det siste designprinsippet som ble beskrevet i kapittel 5.2.2 var utvidbarhet. Utvidbarhet i denne sammenheng tar utgangspunkt i ønsket om å spesifisere en metode som er tilstrekkelig generell til at den kan benyttes i forskjellige applikasjonstyper. Multinodestreaming er en teknikk som har som mål å forbedre tjenestekvalitet, og legger i utgangspunktet ikke føringer på hva selve applikasjonen brukes til. Å gjøre metodespesifikasjonen generell og beskrivelsen av et totalsystem utvidbart har derfor vært et viktig mål. Det har i den foregående spesifikasjonen verken vært lagt føringer på hvilke eksterne komponenter, formater eller liknende som systemet skal benytte. I stedet har vi spesifisert et modulbasert rammeverk uten å si noe om modulenes interne struktur. Eksempelvis har det ikke blitt lagt føringer på hvilken type overlagsnettverk som benyttes. Utvidbarhet er altså et designprinsipp som har blitt tatt hensyn til i metodespesifikasjonen.

Funksjonelle egenskaper

I kapittel 5.3 utdypet vi hvilke funksjoner som inngår i nodeadministrasjonslaget og dermed i vår metodespesifikasjon. Denne funksjonaliteten ble beskrevet på følgende måte:

- Systemet må kunne å motta flere mediestrømmer samtidig.
- Systemet må kunne synkronisere ulike mediestrømmer i forhold til hverandre.
- Endringer i omgivelsene som påvirker ressurstilgjengelighet skal føre til tiltak som kan opprettholde eller gjenopprette tilgjengeligheten hvis dette er mulig.

For disse punktene er det bare funksjonaliteten orientert rundt selve nodeadministrasjonen som er sentral, ikke teknikker som brukes for å utføre nettverkskommunikasjon, bufferhåndtering eller forskjellige medieformater.

Multinodestreaming, det vil si å motta flere mediestrømmer samtidig, er det funksjonelle utgangspunktet for metoden. I spesifikasjonen har det blitt lagt opp til at hver delstrøm kan være i ulike tilstander etter at mottaket har startet. Dermed kan hvert enkelt strømmottak tilpasse sin tilstand ut fra hvilken situasjon den befinner seg i.

Synkronisering av ulike delstrømmer er tildels ivaretatt av nodeadministrasjonslaget ved at en implementasjon kan ha kontrollerte/samkjørte overganger mellom tilstandene før selve mottaket begynner. Denne samkjøringen legger til rette for at oppstarten av hver strøm kan synkroniseres. I tillegg er det mulig å spesifisere en tidsforsinkelse ved start av mottak slik at en ny delstrøm

synkroniseres i forhold til allerede etablerte delstrømmer. Det utføres ingen synkronisering utover dette i selve metoden.

Det siste punktet omhandler gjenoppretting av datatilgjengelighet ved feil. Dette punktet er implementert i form av administrasjon av reservenoder, overvåking av innkommende delstrømmer og unntaksaktiviteter for å gjenopprette leveransen av en strøm så hurtig som mulig.

De funksjonelle egenskapene som hører inn under nodeadministrasjonslaget er altså ivaretatt i metodespesifikasjonen som vi presenterte tidligere i kapittel 5.3, samtidig som viktige designprinsipper er overholdt.

Avhengigheter i forhold til omgivelsene

Avhengigheter mellom en metode og dens omgivelser regner vi som eksterne faktorer som påvirker metodens utførelse. Når en metode skal vurderes isolert sett er det viktig å være oppmerksom på hvilke eksterne faktorer som kan påvirke resultatene av den interne funksjonaliteten. Slike avhengigheter bidrar blant annet til å gjøre testresultater mer usikre, men er samtidig uunngåelige fordi en enkelt funksjon skal fungere i en større kontekst og må derfor ta i bruk eksterne tjenester, samt operere i ikke-kontrollerbare omgivelser.

Nettverksmodellen, illustrert i figur 5.3, viser de forskjellige nettverkskomponentenes plassering i forhold til hverandre. Et lag er avhengig av alle underliggende lag for å fungere. I applikasjonslaget befinner nodeadministrasjonslaget seg på toppen av overlagsnettverket. Nodeadministrasjonslaget er dermed avhengig av at overlagsnettverket fungerer for å fungere selv. Effektiviteten til overlagsnettverket kan dermed også påvirke effektiviteten til nodeadministrasjonslaget, som en direkte følge av at det utføres synkrone kall til overlagsnettverket for å finne informasjon om tilgjengelige noder i nettverket. Fordi det er vanskelig å forutsi effektiviteten til overlagsnettverket når dette ikke er spesifisert, vil det også være vanskelig å nøyaktig måle eller vurdere effektiviteten til nodeadministrasjonsmetoden.

I tillegg til de lavereliggende tjenestene i nettverksmodellen, er egenskaper knyttet til andre noder i nettverket viktige faktorer som kan påvirke resultatene til metoden. Disse egenskapene inkluderer blant annet:

- **Replikasjonsgrad for bestemt objekt:** Antall noder som er i besittelse av et objekt påvirker den faktiske ressurstilgjengeligheten i nettverket, noe som naturligvis påvirker den opplevde ressurstilgjengeligheten. Målet med nodeadministrasjonslaget er å minimere gapet mellom faktisk- og opplevd ressurstilgjengelighet, og den faktiske ressurstilgjengeligheten er dermed en øvre grense for metodens suksess/effektivitet.
- **Andre P2P-noders evne til å levere et objekt:** Hvis en node ikke

er i stand til å levere en strøm, påvirker dette gapet mellom den faktiske- og opplevde ressurstilgjengeligheten i nettverket. En nodes tilgjengelige ressurser (for eksempel prosessor- eller nettverksressurser) kan blant annet påvirke leveringsevnen. Hvis en node utgir seg for å være i stand til å levere et objekt, og at det senere viser seg at dette ikke stemmer, vil det påvirke ytelsen til nodeadministrasjonsmetoden.

- **Ustabile nettverksforhold:** Ustabile nettverksomgivelser kan bidra til store forsinkelser eller pakketap. Slike faktorer påvirker også gapet mellom faktisk- og opplevd ressurstilgjengelighet. Fordi varierende nettverksforhold er en av faktorene som den spesifiserte metoden har som mål å håndtere, regnes den ikke som en feilkilde.

Det er altså mange faktorer (sannsynligvis også flere enn de som er nevnt her) som påvirker metoden. Fordi flere av disse faktorene er vanskelige å kontrollere, vil det være en viss grad av usikkerhet knyttet til testing og analyse av metoden. Det skal imidlertid være en viss grad av usikkerhet for å kunne teste metoden under realistiske omgivelser, og dette er da vanskelig å utføre analytisk. I neste kapittel vil vi derfor beskrive en prototyp som vi har implementert som både et bevis for at konseptet knyttet til multinodestreaming fungerer, og som et referansesystem for tester og analyse. Denne prototypen kan også brukes for å besvare evalueringskriteriene 5 og 6 fra kapittel 5.4.1.

Kapittel 6

Prototyp

Hensikten med å implementere en prototyp er å demonstrere at konseptet knyttet til multinodestreaming og nodeadministrasjon fungerer i praksis, og å være i stand til å utføre beregninger og målinger på dens effektivitet. Prototypen må derfor inkludere nødvendig funksjonalitet for oppsett, leveranse og reetablering av mediestrømmer slik at streamingen kan demonstreres og nødvendige målinger kan utføres.

6.1 Forenklinger

Implementasjonen er utført med enkelte forenklinger som skyldes enten tidsmessige eller tekniske årsaker. Vi har begrenset systemet til kun å håndtere video, og dette ble gjort fordi video enkelt kan deles inn i uavhengige delstrømmer selv om medieformatet som benyttes ikke støtter adaptivitet. Siden det foreløpig ikke er tilgjengelige utviklingsbiblioteker for medieformat med støtte for adaptivitet, som beskrevet i kapittel 4.3, har vi benyttet et ordinært videoformat. For å simulere adaptivitet deles en videostrøm i fire delstrømmer hvor hver delstrøm representerer en kvadrant av videoen. Selv om brukeropplevelsen blir vesentlig dårligere når medieformatet ikke støtter adaptivitet, vil denne mangelen ikke påvirke muligheten for å utføre målinger av mottak og reetablering av mediestrømmer i vesentlig grad.

En annen begrensning er at det ikke er mulig å overvåke strømmer for å oppdage om en avsender slutter å sende data. I stedet simuleres en feilsituasjon ved at brukeren hos mottaker aktivt velger hvilke sendernoder som skal feile. Dette er selvsagt ikke akseptabelt i et reelt system, men for vår prototyp spiller dette en mindre rolle. Simulering av nodefeil kan innebære at det tar kortere tid fra en feil har oppstått til systemet registrerer feilen i forhold til om systemet må oppdage feilen selv. Årsaken til at overvåkningen ikke kan

utføres er sannsynligvis en mangel/feil i Java Media Framework (fork. JMF, se kapittel 6.2.1). Problemet er at hendelsene som skal postes av JMF når en feilsituasjon oppstår aldri blir postet, og dermed er det ikke mulig å oppdage at en sendernode har feilet.

En annen forenkling er at systemet ikke tar høyde for ressursreservasjon hos avsender, og i stedet aksepterer en avsender alle forespørsler som mottas. Dette har imidlertid ikke bydd på problemer ettersom vi ikke har hatt mulighet til å teste systemet med et stort antall noder. Siden antall mottakere var begrenset, ble ikke overbelastning av sendernoder et problem.

For enkelhets skyld ble overlagsnettverket implementert som en sentralisert indekstjener, og ble kjørt på en node som ikke deltok i P2P-nettverket. En sentralisert indekstjener vil ha problemer med å skalere, men testene vi har mulighet til å utføre kommer ikke i nærheten av å inkludere mange nok noder til at effekten av skalerbarhetsproblemet merkes.

Vi har ikke hatt mulighet til å teste systemet i varierende omgivelser. Alle tester er utført under tilnærmet homogene forhold. Alle noder vi har hatt tilgang til befinner seg på NTNU sitt nettverk hvor den tilgjengelige båndbredden er høy og forsinkelsen er lav. En annen forenkling vi har gjort er å sette forsinkelsen mellom noder ved hjelp av en tilfeldighetsfunksjon. Forsinkelsen mellom to noder brukes av mottaker for å rangere potensielle avsendere, men siden forsinkelsen mellom alle noder vi har benyttet i tester er svært lav ville det uansett vært vanskelig å rangere potensielle avsendere etter forsinkelse. Årsaken til at forsinkelsen simuleres i stedet for å måles er at utviklingsbibliotekene som ble benyttet ikke har støtte for protokollen "Internet Control Message Protocol" (ICMP) som vanligvis brukes for å måle forsinkelse.

6.2 Arkitektur og implementasjonsbeskrivelse

Den implementerte prototypen tar utgangspunkt i den overordnede arkitekturen beskrevet i kapittel 5.2.3. Den overordnede arkitekturen ble blant annet illustrert som et moduldiagram (se figur 5.4) som gir en oversikt over den konseptuelle funksjonaliteten i et totalsystem. I dette kapittelet vil vi utdype de arkitektoniske valgene som har blitt gjort i forhold til implementasjonen av en prototyp. Vi vil også legge vekt på å beskrive hvordan konseptuell funksjonalitet har blitt realisert.

6.2.1 Teknologivalg

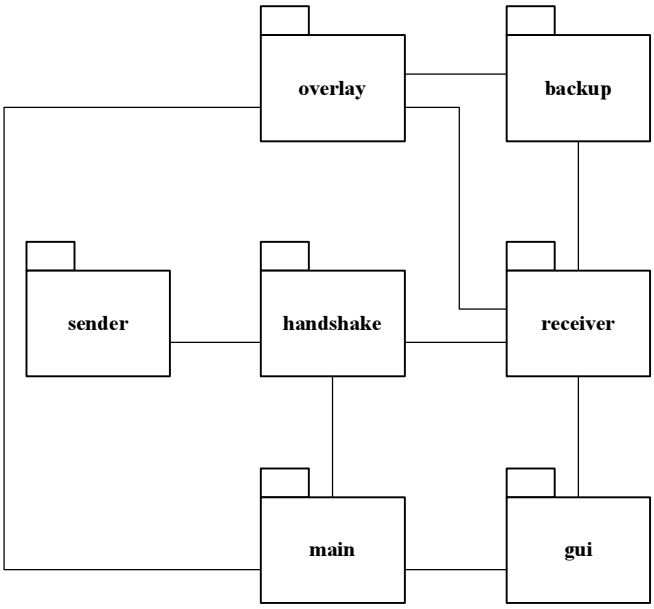
Prototypen er implementert i programmeringsspråket Java. Til dette har vi brukt "Java 2 Platform, Standard Edition" (J2SE), versjon 1.4.2. J2SE inklu-

derer blant annet et sett av standardbiblioteker som støtter utviklingen av Java-baserte applikasjoner. I tillegg til standardbibliotekene har vi benyttet Java Media Framework (JMF) [Sun Microsystems Inc., 2004] for å bygge multimediefunksjonaliteten i prototypen. JMF muliggjør at lyd, video og andre tidsavhengige medier kan benyttes i Java-baserte applikasjoner. JMF inkluderer blant annet biblioteker som muliggjør nettverksleveranse og -mottak av digitale medier, samt koding og avspilling av disse. JMF har blant annet støtte for nettverksprotokollen RTP (se kapittel 3.4) for en rekke medieformater. I prototypen har vi valgt å bruke formatet MJPEG ("Motion-JPEG") for å kode video, og transporterer en generert MJPEG-strøm over RTP.

I nodeadministrasjonsmetoden, som vi spesifiserte i forrige kapittel, ble det utvekslet en rekke meldinger mellom noder i forkant av leveransen av selve mediestrømmen. I prototypen er denne meldingsutvekslingen implementert ved hjelp av Java RMI ("Remote Method Invocation") som inngår i standardbiblioteket til Java. RMI muliggjør metodekall mellom systemer over nettverk. Ved å bestemme et grensesnitt mellom to noder, er det mulig å kalle metoder hos den andre noden ved hjelp av RMI. RMI er gunstig i forhold til å utveksle informasjon i forkant av strømleveranse fordi den tillater synkrone kall over nettverk. En node som ønsker å etablere en sesjon med en annen node, vil i vår prototyp kalle en forespørselsmetode hos den andre noden. Metoden vil senere returnere med resultatet av forespørselen. RMI er også gunstig fordi den skjuler mye av kompleksiteten knyttet til metodekall over nettverk. RMI er også robust i forhold til å håndtere feil som kan skje både på nettverket og hos den enkelte node.

6.2.2 Komponenter/pakker

I prototyp-implementasjonen har vi valgt å innkapsle relatert funksjonalitet i pakker. En pakke er en konseptuell enhet som representerer en komponent eller hovedfunksjon i systemet. Et pakkediagram over systemet er illustrert i figur 6.1. En pakke i det implementerte systemet kan bestå av flere separate funksjoner, og det er derfor ikke noen én-til-én kobling mellom pakkene og modulene i det konseptuelle pakkediagrammet over arkitekturen (figur 5.4).



Figur 6.1: Pakker i systemet (UML pakkediagram).

Pakkene som prototypen er delt opp i er som følger:

overlay Overlagsnettverket, samt funksjonaliteten for å kommunisere med overlagsnettverket, er implementert i pakken overlay. Denne pakken tilsvarer modulen "Overlag" i den overordnede arkitekturen fra kapittel 5.4. Selve P2P-applikasjonen benytter kun denne pakkens grensesnitt mot overlagsnettverket, og ikke selve klassene som implementerer overlagsnettverket. Overlagsnettverket realiseres som en sentralisert indekstjener, og kjører som en separat applikasjon på én separat node i nettverket.

receiver Funksjoner som i arkitekturen finnes i modulen "Avspilling", i mottaksdelen av modulen "Strømtransport", i modulen "Strømkontroll" og i modulen "Bufring og synkronisering", er i implementasjonen plassert i pakken receiver. Klassene i receiver tar seg av mottak, koordinering, synkronisering og dekomprimering av mediestrømmer, og pakken inneholder i tillegg funksjonalitet for å tegne video på skjermen. Funksjonaliteten for å avgjøre hvilke noder et objekt skal hentes fra (og dermed også å rangere noder i forhold til hverandre) finnes også i denne pakken.

sender Senderdelen av modulen "Strømtransport" i den overordnede arkitekturen representeres i prototypen som pakken sender. Funksjonaliteten i denne pakken brukes følgelig til å levere mediestrømmer til mottakernoder.

handshake Forespørsel, forhandling og etablering hos både avsender og mottaker utføres i pakken handshake, og denne pakken svarer derfor til modulen "Strømoppsett" i den overordnede arkitekturen. Datastrukturer som utveksles mellom noder under forhandling og etablering inngår også i denne pakken.

backup Feilhåndteringsdelen av modulen "Strømkontroll" i den overordnede arkitekturen er implementert som pakken backup. En liste over potensielle avsendere opprettes og vedlikeholdes her slik at en mediestrøm som går tapt raskt kan reetableres med en ny avsender.

gui Brukergrensesnittet finnes i pakken gui, og tilsvarer modulen "Brukergrensesnitt" i arkitekturen.

main Selve P2P-applikasjonen (ikke overlagsnettverket) startes fra pakken main, og denne håndterer også alle konfigurasjonsparametre som enten er spesifisert i kildekoden eller leses inn fra fil.

I prototypen vil en node kunne ha to roller:

- Mottaker
- Avsender

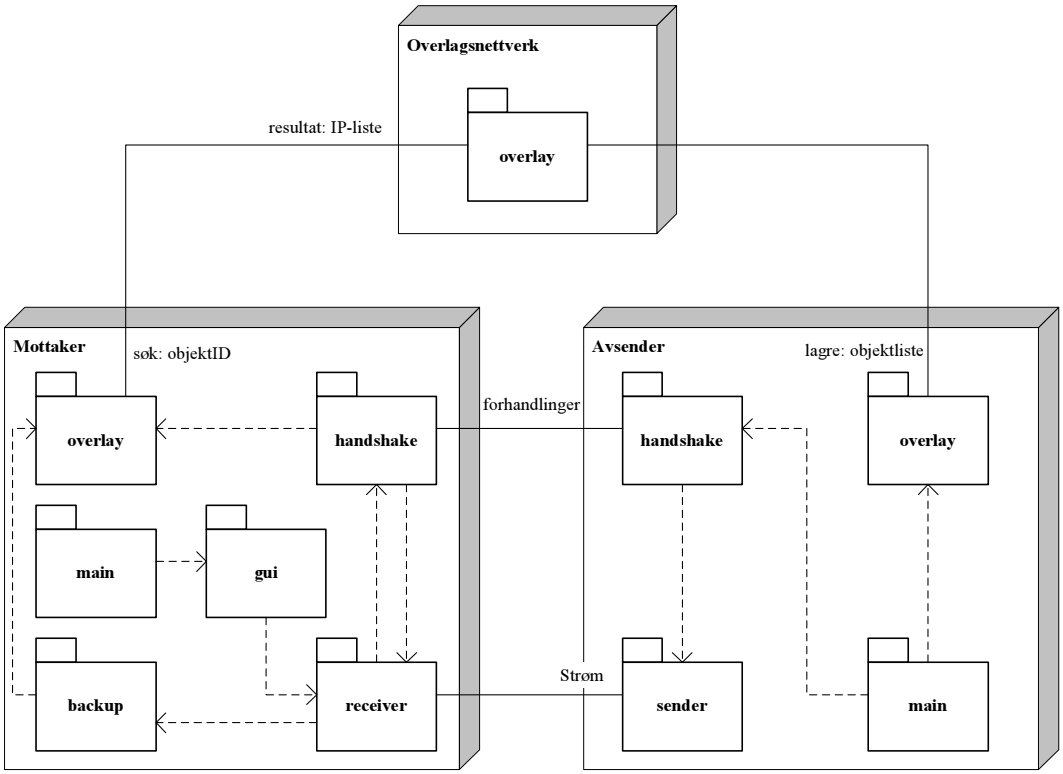
I tillegg vil en node ha rollen som overlagsnettverk.

Som beskrivelsen av de forskjellige pakkene viser, er det en viss blanding av roller innen hver pakke. Pakken "handshake" inneholder eksempelvis både funksjonalitet for å starte forhandlinger på mottakersiden og å svare på henvendelser på avsendersiden. Dette er imidlertid helt naturlig fordi en node kan ha flere roller. Samtidig som applikasjonen leverer en delstrøm til en annen node, kan brukeren av applikasjonen hente flere delstrømmer fra andre noder i nettverket. Implementasjonsmessig har det derfor ikke blitt lagt vekt på å adskille roller i form av pakker. Funksjonsmessig er det imidlertid en klar rolledeling, noe som er videre beskrevet i neste avsnitt.

6.2.3 Distribusjon

I prototypen har vi gjort en forenkling som gjør at overlagsnettverket implementeres som en separat applikasjon som kjører på en dedikert node. Denne applikasjonen fungerer da som en sentralisert indekstjener som svarer på objektforespørsler fra andre P2P-noder. Overlagsrollen er derfor skilt ut fra resten av applikasjonen.

Samtlige noder i P2P-nettverket (hvis vi ser bort fra overlagsnoden) har en delt funksjon, nemlig både å motta og levere data. For en bestemt sesjon (tjenesteavtale med en annen node) har imidlertid en node kun én funksjon/rolle. Derfor vil en node, for en bestemt sesjon, enten være avsender eller mottaker av en delstrøm. Som mottaker vil en node ofte ha opprettet flere sesjoner for et objekt (fordi et objekt leveres som flere delobjekter fra forskjellige noder). Figur 6.2 illustrerer en nodes funksjon i forhold til én sesjon. I figuren er det en overlagsnettverksnode, en mottakernode og en avsendernode. Hver node er illustrert som en kube, mens pakker er illustrert på tilsvarende måte som i figur 6.1.



Figur 6.2: Distribusjon av funksjonalitet (UML deployment-diagram).

Formålet med figuren er å vise hvordan funksjon er knyttet til en bestemt rolle og hvordan funksjonen er distribuert på ulike noder. I figuren er det opprettet en sesjon mellom to av nodene, hvor noden til venstre opptrer som mottaker og noden til høyre opptrer som avsender. I tillegg illustrerer figuren hvilke pakker fra prototypen som er involvert for en bestemt funksjon. Pilene mellom pakkene viser hvilke pakker som oppretter eller har kontroll over funksjonalitet i andre pakker. En pil peker fra en pakke som har kontroll over funksjonalitet i en annen pakke (pilen peker mot denne).

Nodene i figur 6.2 utfører blant annet følgende funksjoner knyttet til sin rolle:

Mottaker Mottakernodens funksjonalitet starter i pakken main. Main har ansvar for å lese eksterne innstillinger og å starte det grafiske brukergrensesnittet (representert som pakken gui). Gjennom brukergrensesnittet kan brukeren fortelle systemet hvilket objekt som ønskes mottatt. Når dette er spesifisert, kan selve strømmottaksfunksjonaliteten opprettes, i form av pakken receiver. For å kunne motta data er det først nødvendig å finne ut hvilke noder som er i besittelse av det ønskede objektet, og deretter starte forhandlinger med et antall avsendernoder. Funksjonaliteten i pakken handshake tar seg av forhandlinger, og bruker resultatene fra pakken overlay til å finne ut hvilke noder den kan kontakte. Når forhandlinger er utført (og har lykkes), er mottakernoden klar til å motta data. Dette gjøres ved hjelp av strømmottaksfunksjonaliteten i pakken receiver. Receiver starter samtidig reservenodeadministrasjonen, representert som pakken backup. Denne pakken benytter funksjoner i pakken overlay til å jevnlig spørre overlagsnettverket slik at den kan holde orden på hvilke noder som er i stand til å levere det ønskede objektet.

Avsender Avsendernodens funksjonalitet starter også i pakken main. Under oppstart av applikasjonen brukes funksjoner i pakken overlay til å kommunisere med overlagsnettverket for å fortelle hvilke objekter noden kan levere. Deretter begynner avsendernoden å lytte etter henvendelser fra andre noder i nettverket. Dette gjøres ved å starte forhandlingsfunksjonaliteten i pakken handshake. Når avsendernoden mottar en forespørsel, utføres de nødvendige forhandlingene med den forespørrende noden. Hvis forhandlingene lykkes, startes funksjonaliteten for å streame det aktuelle objektet over nettverket. Denne funksjonaliteten finnes i pakken sender. Sender kommuniserer med mottaksfunksjoner i pakken receiver hos mottakernoden.

Overlagsnettverk Overlagsnettverket kjører som et separat system på en dedikert node. Dens oppgaver er å holde orden på hvilke noder som tilbyr hvilke objekter, og svare på spørsmål om hvilke noder som kan levere et bestemt objekt. Funksjonaliteten for overlagsnettverket finnes i pakken overlay. En hvilken som helst node kan i prinsippet fungere som overlagsnettverk.

Slik prototypen er implementert vil samtlige noder kunne påta seg rollen som avsender fra det tidspunktet den melder seg inn i nettverket. Om en node kommer til å levere data avhenger blant annet av hvilke objekter som blir forespurt, samt om noden blir rangert høyt nok til at andre noder ønsker å bruke den som avsender. Sistnevnte betingelse er en form for svakhet i systemet (i forhold til en ideell P2P-arkitektur) fordi den til en viss grad kan bidra til å konsentrere last på ressurssterke noder. I verste fall kan man risikere at

en node ikke har tilstrekkelige ressurser til å levere et delobjekt. En løsning på dette problemet er å kreve at en node har et minimum av ressurser for å selv kunne nyttegjøre andres ressurser i nettverket. Det er imidlertid urealistisk å kreve like noder (det vil si med like prosessor-, lagrings- og nettverksressurser) så lenge P2P-nettverket ikke opererer under kontrollerte omgivelser. Ellers har vi for hele nettverket antatt at det kun finnes én node som opererer som overlagsnettverk, mens antall mottakere avhenger av mengden objektforespørsler.

I neste kapittel vil fokus være å beskrive hvordan prototypen fungerer sett fra brukerens perspektiv. For å forstå prototypens virkemåte er blant annet forståelsen av forholdet mellom indre funksjonalitet og det ytre brukergrensesnittet viktig. Videre vil vi fokusere på å teste prototypen opp mot de evalueringskriteriene fra kapittel 5.4 som ennå ikke er vurdert.

6.3 Funksjonalitet og virkemåte

Vi har nå beskrevet prototypens oppbygning og arkitektur, blant annet i form av dens oppdeling i konseptuelle pakker og hvilke roller systemet kan ha i et P2P-nettverk. For hver pakke og rolle har vi også beskrevet den overordnede funksjonaliteten som ligger bak, med fokus på å beskrive parallellene til nodeadministrasjonsmetoden som ble spesifisert i kapittel 5. I dette delkapittelet vil vi rette fokus mot hvilke funksjoner prototypen tilbyr brukeren, og hva som forgår i P2P-nettverket når systemet er i bruk. I den forbindelse vil vi blant annet presentere en rekke skjermbilder av brukergrensesnittet som tilbys brukeren av prototypen. Målet med demonstrasjonen er å vise hvordan prototypens funksjonalitet og virkemåte tar seg ut sett fra brukerens perspektiv, samt for å vise prototypens kapabiliteter.

For å demonstrere funksjonaliteten ble et lite P2P-nettverk med 13 noder satt opp. Samtlige noder var i utgangspunktet i stand til å levere de samme objektene (og dermed også de samme delobjektene). Én av nodene var mottaker, mens de resterende nodene var avsendere. Fordi det er mottakernoden som utfører nodeadministrasjon (for sine streamingsesjoner) og spiller av selve medieobjektet, er det mottakerens funksjon som er mest interessant å teste og demonstrere. I tillegg til vanlige P2P-noder, kjørte vi et overlagsnettverk på en separat node. Medieobjektet som ble brukt til streaming og avspilling er en trailer til animasjonsfilmen “Shrek 2”, som blant annet er tilgjengelig fra [Apple Computer, Inc, 2004].

Kjøringen av applikasjonen kan deles i tre hovedfaser:

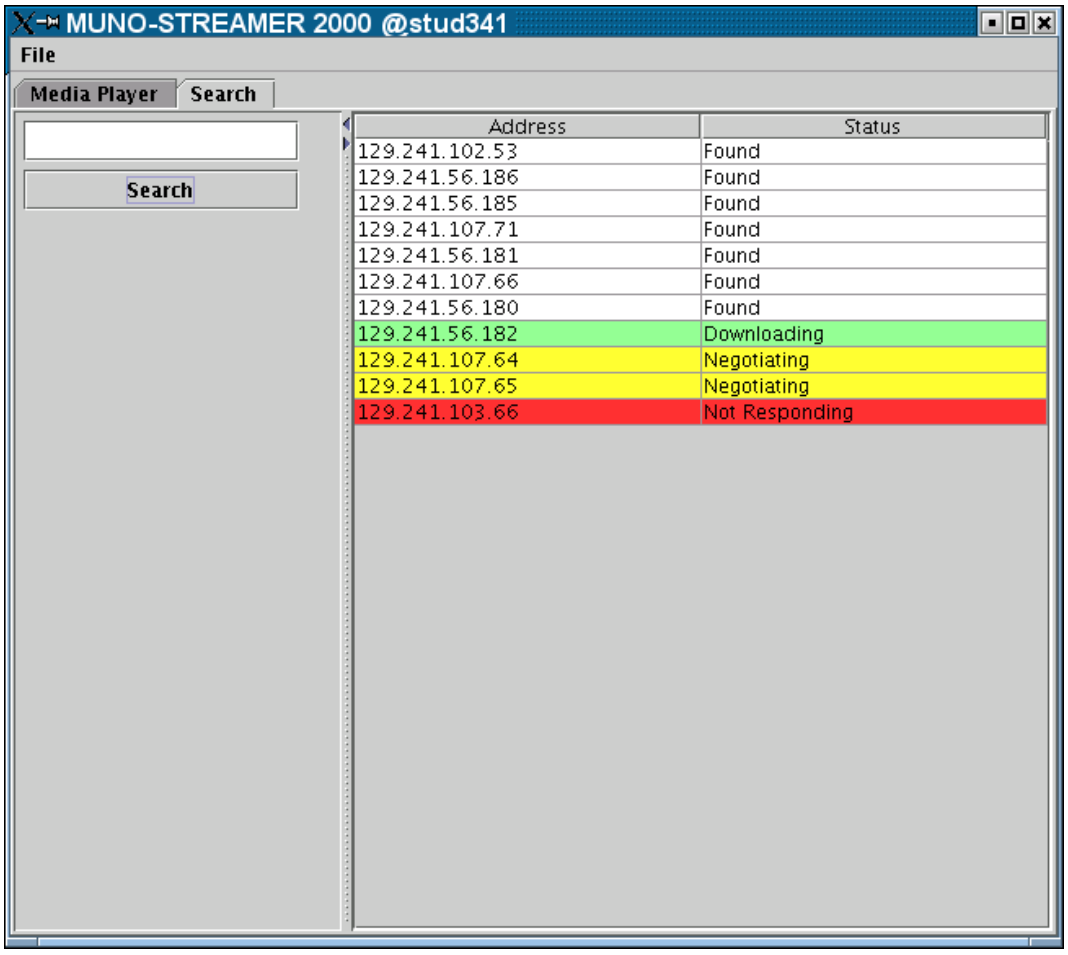
- Søk og forhandling
- Streaming og avspilling

- Feilhåndtering

I de følgende avsnittene vil hver av disse fasene bli presentert og forklart.

6.3.1 Søk og forhandling

En streamingsesjon begynner med at brukeren spesifiserer hvilket objekt som ønskes levert. Når dette er gjort er det systemet selv som vurderer hvilke og hvor mange noder den ønsker å hente objektet fra. Brukeren kan følge med på denne prosessen i brukergrensesnittet. Figur 6.3 viser et skjermbilde av brukergrensesnittet som viser de foreløpige søkeresultatene, samt status for hvert resultat, for det gjeldende objektet.



Figur 6.3: Søkeresultater og pågående aktiviteter.

Brukergrensesnittet er delt i en søkedel og en avspillingsdel. Figur 6.3 viser

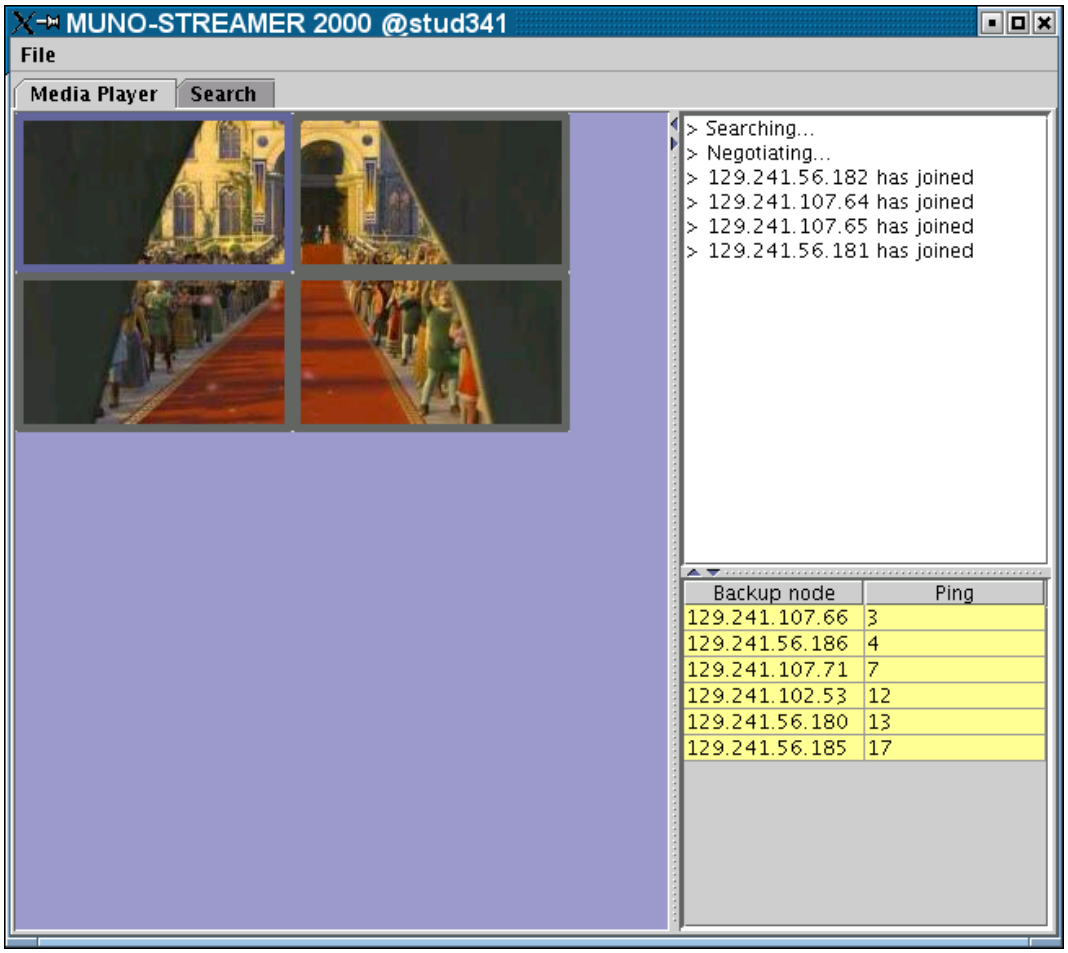
søkedelen av brukergrensesnittet. Denne er igjen oppdelt i to deler. Den ene delen, lokalisert til venstre i skjermbildet, tar imot data fra brukeren i form av identifikatoren til det ønskede medieobjektet. Den andre delen, lokalisert til høyre, viser de foreløpige søkeresultatene fra overlagsnettverket i form av en liste over IP-adressene til noder som kan levere objektet. Hver node har en status som er angitt i høyre kolonne av tabellen. I figuren har én av nodene ikke svart på objektforespørselen og har derfor status "Not Responding" (svarer ikke). Det er innledet (men ikke avsluttet) forhandlinger med to av nodene, og disse har status "Negotiating" (forhandler). En fjerde node har akseptert forespørselen og har allerede begynt å levere mediedata. Denne noden har derfor status "Downloading" (laster ned).

Innstillingene i prototypen for denne demonstrasjonen tilsier at streaming og avspilling kan starte med kun én avsendernode. Denne innstillingen kan tilpasses etter hva som aksepteres som minimum avspillingskvalitet. Hadde grensen vært satt til to delobjekter, ville ikke avspillingen startet før et tilstrekkelig antall noder kunne levere data. Når en ny node er klar til å streame data til mottakernoden, må den synkronisere sitt interne tidsbegrep med de øvrige nodene. Det er mottakernoden som har ansvaret for å utføre all avspillingssynkronisering.

6.3.2 Streaming og avspilling

Figur 6.4 viser avspillingsdelen av prototypen. Skjermbildet er tatt etter at fire noder har begynt å levere mediedata, slik at maksimal avspillingskvalitet er oppnådd. Selve avspillingen av videoen foregår øverst til venstre i brukergrensesnittet. Avspillingsvinduet er delt i fire deler som tydelig viser hvert delobjekt som leveres.

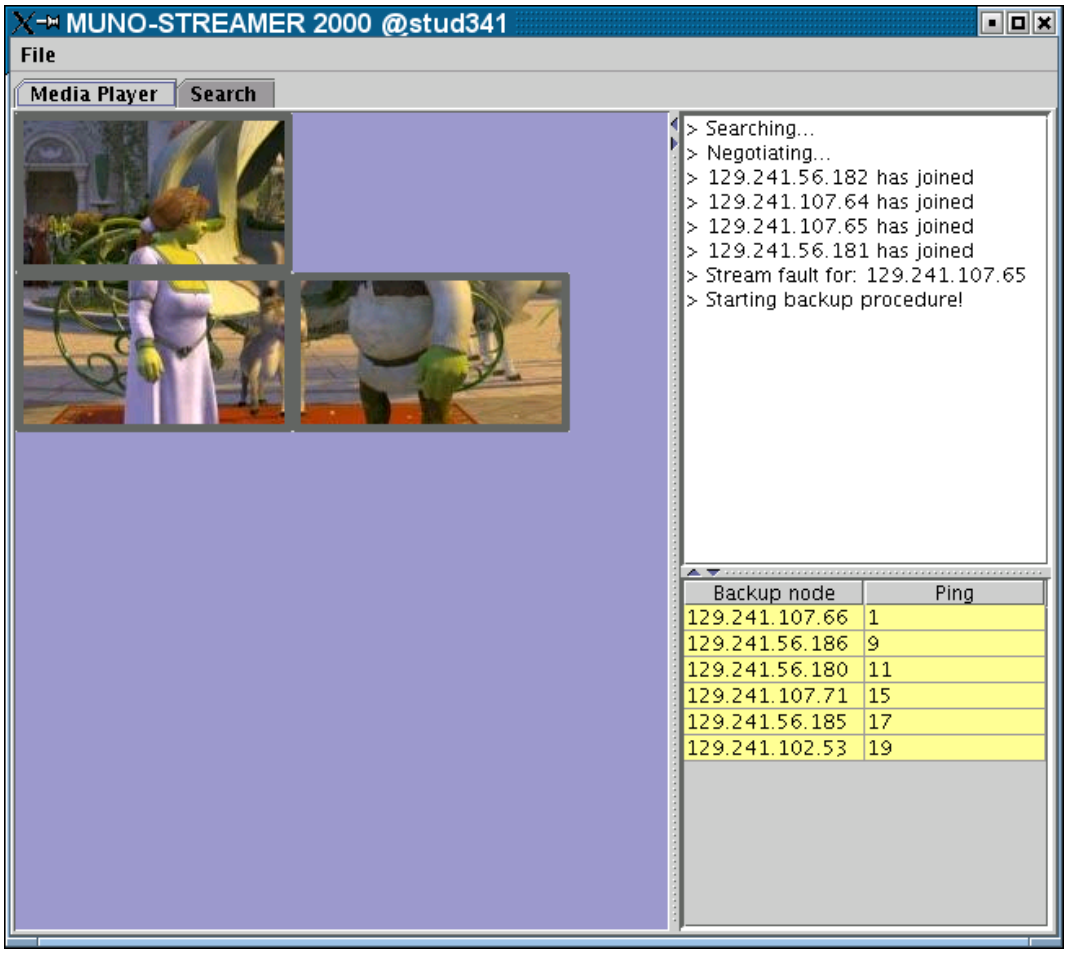
Under avspilling vil applikasjonen kontinuerlig oppdatere en prioritert liste over potensielle reservenoder som kan brukes hvis en feil oppstår. Denne listen vises som en tabell nederst til høyre i skjermbildet i form av nodenes IP-adresser og generert ping-verdi (forsinkelse i millisekunder). Informasjon om hendelser som påvirker avspillingen av medieobjektet skrives ut i logg-feltet øverst til høyre.



Figur 6.4: Streaming og avspilling av fire delobjekter satt sammen til ett bilde.

6.3.3 Feilhåndtering

Feilhåndtering skal i utgangspunktet skje når en node enten slutter å levere data, eller hvis dataene som leveres er av dårlig kvalitet (for eksempel på grunn av stort pakketap). I prototypen er det mulig å fremprovosere nodefeil i brukergrensesnittet til mottakernoden. Dette gjør det blant annet enkelt å teste feilhåndteringsfunksjonen. Figur 6.5 viser avspillingsdelen av brukergrensesnittet etter at en av nodene har feilet (fremprovosert delstrømfeil). Kvaliteten av avspillingen har på dette punktet blitt degradert med 25 %, og feilhåndtering pågår i bakgrunnen i forsøk på å gjenopprette full avspillingskvalitet. Brukeren blir informert om at feilhåndtering pågår i logg-feltet.

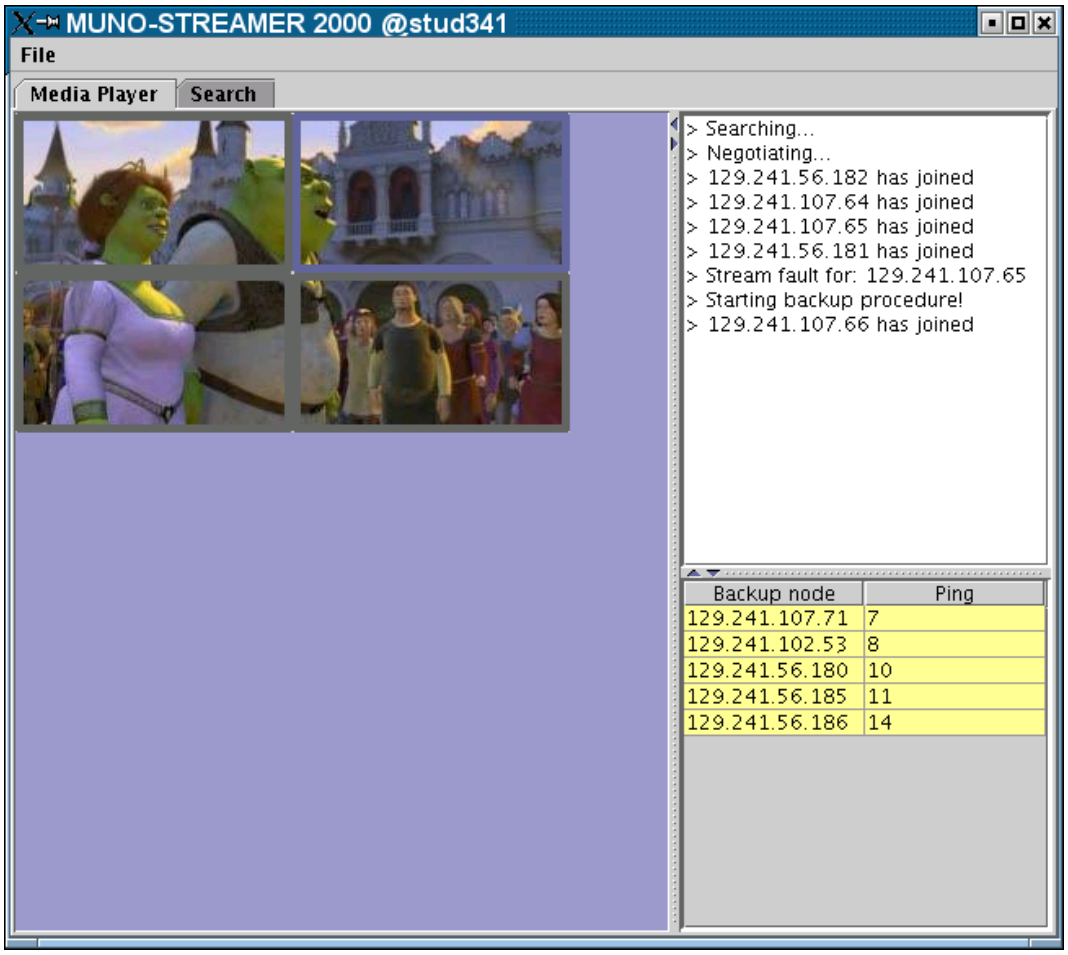


Figur 6.5: En node har feilet. Feilhåndtering pågår i bakgrunnen.

Når en feil oppstår har mottakernoden allerede tatt stilling til hvilken alternativ node den ønsker å hente det manglende delobjektet fra. Dermed kan noden straks sette i gang prosedyren for å opprette leveranse av delobjektet — uten å ta kontakt med overlagsnettverket først. Hvis noden som har høyest rangering av reservenodene ikke svarer eller ikke er i stand til å levere objektet, vil neste node på listen kontaktes. Innen den tid kan reservenodelisten ha blitt oppdatert av bakgrunnsprosessen som stadig sjekker overlagsnettverket for oppdatert informasjon om tilgjengelige ressurser, samt sorterer reservernodelisten ut fra informasjon om forsinkelse.

Mottakernoden inngår umiddelbart forhandlinger med den første reservenoden som svarer positivt på objektforespørselen. Når forhandlingene er gjennomført, starter leveransen av det manglende delobjektet. Denne leveransen synkroniseres i tid med de øvrige objektleveransene slik at det er mulig å spille av det sammensatte medieobjektet ved å spille alle delobjektene i parallell.

Figur 6.6 viser systemet etter at avspilling med full kvalitet er gjenopprettet. Den nye noden har synkronisert sin leveranse etter de øvrige leveransenodene, og systemet har gjenopprettet normaltilstand. Ved dette tidspunktet har systemet fortsatt fem potensielle reservenoder i tilfelle en ny node feiler.



Figur 6.6: Normal avspilling med full kvalitet er gjenopprettet.

I neste delkapittel vil en rekke tester bli utført med det formål å måle hvor effektive de forskjellige funksjonene i prototypen er. Disse testene vil bidra til å beskrive hvor effektivt funksjonaliteten beskrevet til nå kan utføres under kontrollerte omgivelser, og vil også kunne besvare de gjenværende evalueringskriteriene fra kapittel 5.4.

6.4 Testing

Testingen av prototypen tar sikte på å vurdere dens funksjon og effektivitet. Målene med testingen av prototypen er å beregne følgende:

- Den gjennomsnittlige tiden som brukes for å reetablere en tapt delstrøm.
- Den gjennomsnittlige tiden som brukes per fase under reetablering. Reetablering deles i følgende faser:
  - Forhandling.
  - Behandling av forhandlingsresultat.
  - Etablering av strømmottak.
- Den prosentvise avspillingsdegraderingen som funksjon av antall nodefeil per tidsenhet.

I tillegg ønsker vi å kunne vurdere prototypen opp mot evalueringskriteriene som ble fastsatt i kapittel 5.4. Videre i dette delkapittelet vil vi først beskrive hvordan testene ble utført. Deretter vil resultatene fra selve testene bli presentert.

6.4.1 Utførelse

Det ble utført totalt fem uavhengige tester. Samtlige noder i systemet ble nullstilt (ved at prototypen ble startet på nytt) for hver test. Hver test ble utført på tilsvarende måte og under tilsvarende omgivelser. Betingelsene for testutførelsen var:

- Det forespurte objektet består av totalt fire delobjekter med karakteristikker som vist i tabell 6.1.
- Ingen noder kan være i besittelse av kun en delmengde av delobjektene tilhørende et objekt.
- Ingen avsendernoder kan levere mer enn ett delobjekt til samme mottakernode.
- For hver test skal hver delstrøm feile nøyaktig én gang. Dette medfører at mottakernoden må håndtere fire delstrømfeil per test.
- Hver test skal utføres i et P2P-nettverk med totalt 13 noder som er i stand til å levere det forespurte objektet.

- Mottakernoden skal starte avspilling så snart en delstrøm er tilgjengelig. Minimumskravet til avspillingsstart er altså 25 % av full kvalitet.
- For hver feil som genereres skal systemet måle tiden det tar å reetablere den tapte delstrømmen, samt den forbrukte tiden for den enkelte fase i reetableringen.

	Objekt 1	**Objekt 2**	**Objekt 3**	**Objekt 4**
Bitrate	541 Kb/s	517 Kb/s	565 Kb/s	554 Kb/s
Varighet	80 s			
Dimensjon	160x88 (piksler)			
Bilder/sekund	24			
Kodek/filformat	MJPEG/AVI			

Tabell 6.1: Mediekarakteristikker for hvert delobjekt.

Som tabell 6.1 viser, må en mottakernode være i stand til motta data med en båndbredde på minimum 541 + 517 + 565 + 554 = 2177 Kb/s for å kunne motta objektet med full kvalitet. Avspillingsområdet vil da ha en dimensjon på 640x352 piksler. Testdataene stiller altså forholdsvis høye båndbreddekrav til de deltakende nodene i P2P-nettverket. I testingen har vi benyttet noder med maksimal fysisk båndbredde på 100 Mb/s (Ethernet), det vil si med god margin i forhold til kravet for å kunne oppnå full kvalitet. Også i mange kommersielle Internett-løsninger vil det i dag være mulig å motta en strøm med over 2 Mb/s båndbredde (se kapittel 2.3.3), og forholdene ligger altså allerede i dag til rette for å realisere et tilsvarende P2P-nettverk over Internett.

Ved å utføre fem uavhengige tester som hver håndterer fire nodefeil, vil grunnlaget for å beregne metodens effektivitet være til sammen 20 nodefeil. Ved å utføre et større antall tester, vil nøyaktigheten kunne forbedres. Vi anser imidlertid 20 nodefeil for å være tilstrekkelig for å kunne vurdere metodens egenskaper i lys av evalueringskriteriene gitt i kapittel 5.4 og innledningsvis i dette delkapittelet.

6.4.2 Resultater

For hver av de fem testene ble følgende data hentet ut:

- Tidspunktet hvor en hendelse som påvirker avspillingskvaliteten inntreffer. En slik hendelse er at en delstrøm ikke lenger leveres (degradert

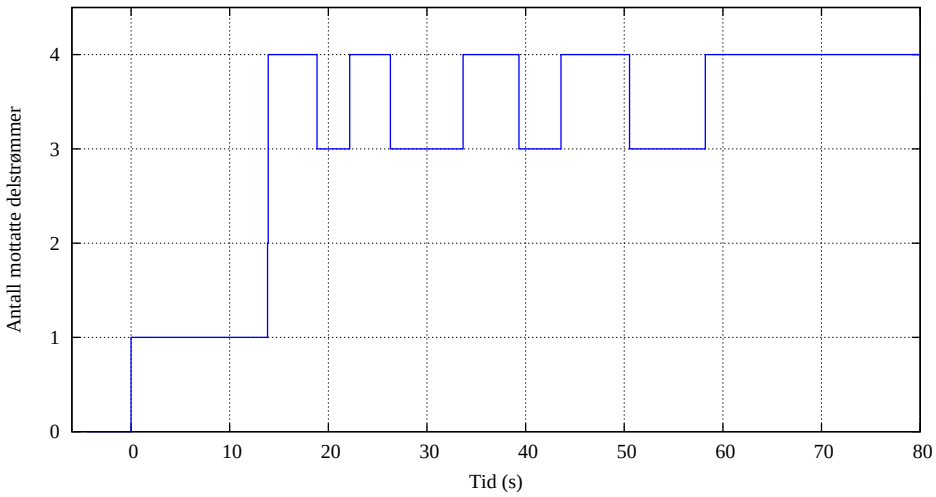
kvalitet) eller at en ny delstrøm mottas (oppgradert kvalitet). Tidspunktet måles i antall millisekunder siden avspillingsobjektet ble forespurt av mottakernoden.

- Start- og sluttidspunktet for hver fase i reetableringsprosedyren.

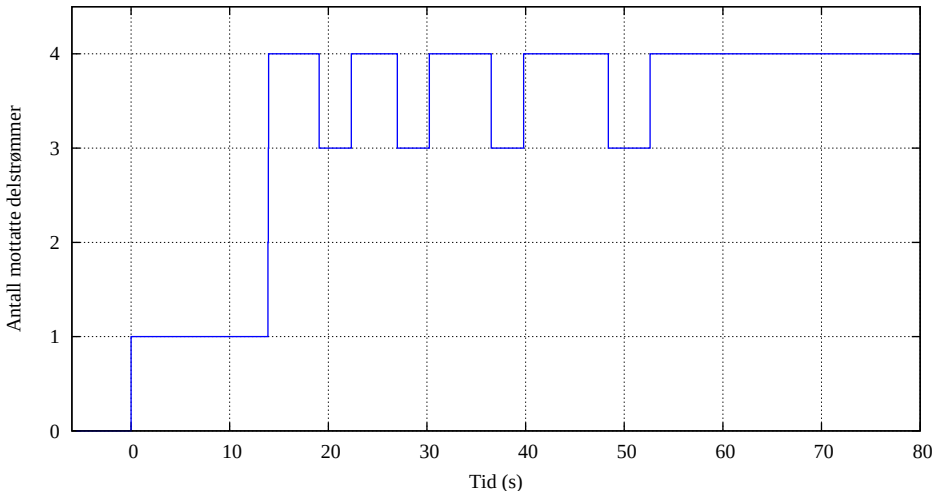
Alle testresultatene er i sin helhet presentert i vedlegg B. I dette delkapittelet presenteres en graf over hendelsesforløpet for hver av testene. Denne grafen illustrerer antall delstrømmer som mottas på et gitt tidspunkt. Antall delstrømmer er som tidligere nevnt et mål på kvaliteten på avspillingen. I tillegg til hendelsesforløpet presenteres beregnede gjennomsnittsverdier for varigheten av hver fase i reetableringsprosedyren.

Grafene i figurene 6.7 – 6.11 illustrerer hendelsesforløpet til testene 1 – 5. Hver av grafene viser hvordan kvaliteten på avspillingen varierer under gjennomføringen av testen. Hver test har først en fase med redusert kvalitet fordi mottakeren ennå ikke har etablert alle fire delstrømmene som kreves for å oppnå full avspillingskvalitet. Denne oppstartsfasen kan unngås ved å endre innstillingene i prototypen slik at minimum antall delstrømmer for å starte avspilling er fire. I testene regner vi ikke med degraderingen i oppstartsfasen fordi den ikke er en følge av delstrømfeil.

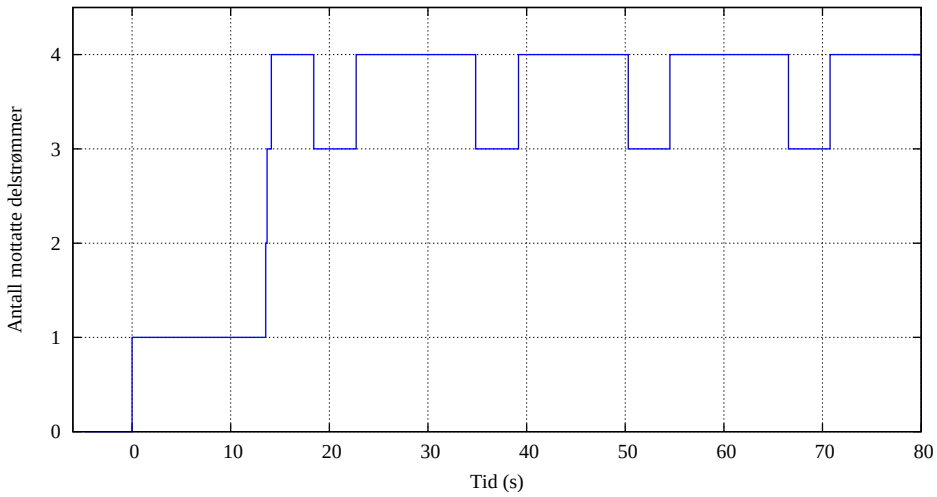
For hver av testene genereres fire delstrømfeil. I grafen vises dette ved at kvalitetsnivået senkes ett nivå. Kvalitetsnivået heves igjen etter at den feilede delstrømmen er reetablert.



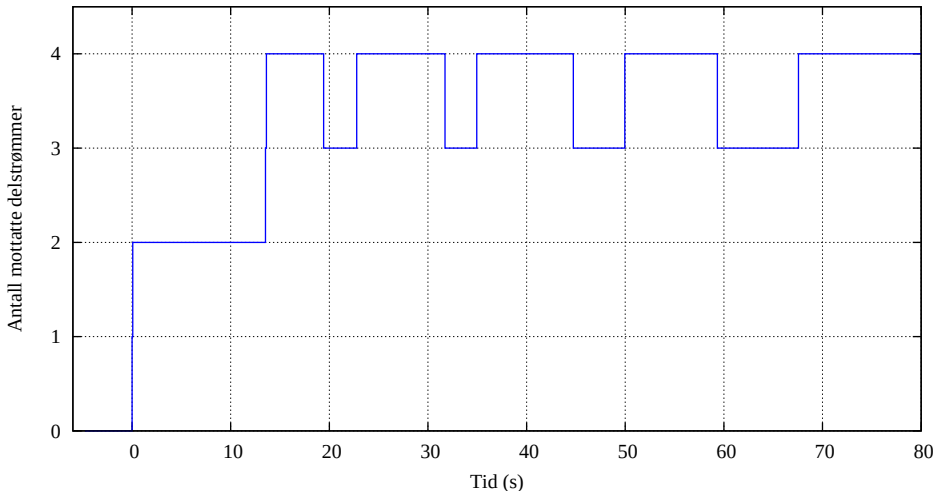
Figur 6.7: Variasjon i antall mottatte delstrømmer for test 1.



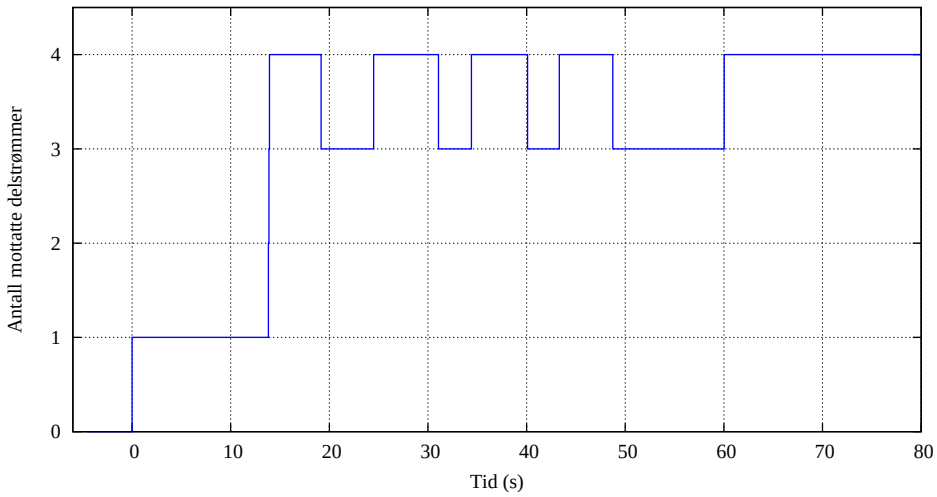
Figur 6.8: Variasjon i antall mottatte delstrømmer for test 2.



Figur 6.9: Variasjon i antall mottatte delstrømmer for test 3.

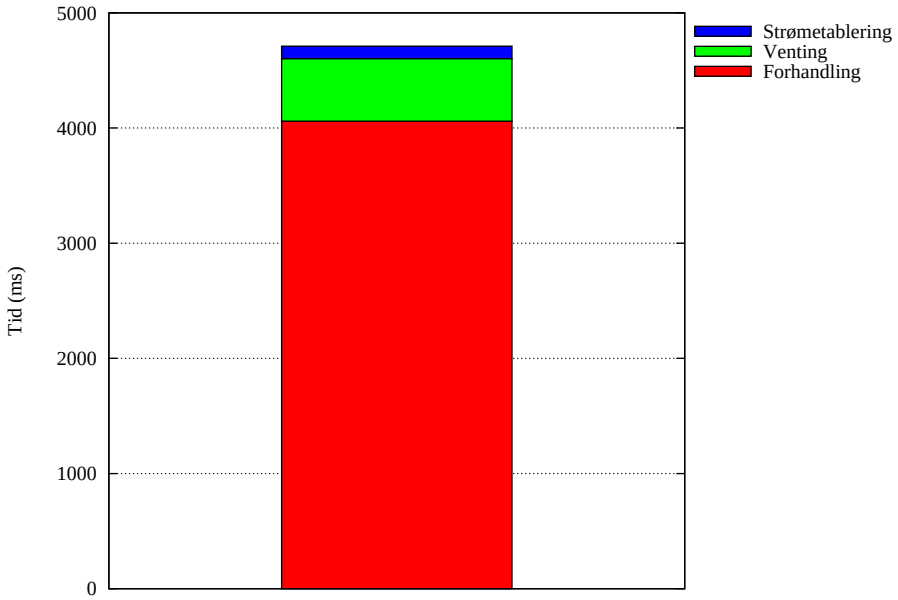


Figur 6.10: Variasjon i antall mottatte delstrømmer for test 4.



Figur 6.11: Variasjon i antall mottatte delstrømmer for test 5.

Søylediagrammet i figur 6.12 viser det gjennomsnittlige tidsforbruket for hver fase under reetablering av en delstrøm. Verdiene, angitt med hver sin farge i diagrammet, er beregnet på grunnlag av samtlige genererte delstrømfeil i de fem testene, altså tilsammen 20 delstrømfeil. Disse gjennomsnittsverdiene gir et bilde av hvilke faser som er kritiske (med tanke på tid) for reetableringens effektivitet. Sammen utgjør fasene den totale tiden som kreves for å reetablere en delstrøm. Kildedataene for diagrammet i figuren finnes i vedlegg B.



Figur 6.12: Gjennomsnittlig tidsforbruk per fase.

Testresultatene vi har oppnådd ved å gjøre målinger på prototypen har enkelte feilkilder og begrensninger. De gir imidlertid et rimelig godt bilde av hvilke kapabiliteter prototypen har, og hvilken effektivitet selve nodeadministrasjonsmetoden kan oppnå. I neste delkapittel vil vi evaluere testresultatene — blant annet opp mot de gjenværende evalueringskriteriene fra kapittel 5.4.

6.5 Vurdering av prototypen

Målet med nodeadministrasjonslaget i nettverksmodellen var å muliggjøre multinodestreaming og effektiv reetablering av delstrømmer som feiler. Målet med multinodestreaming var å gjøre konsekvensen av at en node feiler så lav som mulig for avspillingen av et medieobjekt. Fordi dataene fra én avsendernode kun utgjør en delmengde av de dataene som totalt spilles av hos mottakernoden, vil en enkelt nodefeil få mindre konsekvenser enn om avsendernoden hadde ansvaret for hele leveransen. Med kun én strøm per medieobjekt ville mottakernoden opplevd et totalt strømbrudd og stans i avspillingen. Med multinodestreaming kan avspillingen fortsette, men med degradert kvalitet. Med enkeltnodestreaming vil et totalt strømbrudd kunne medføre at den opplevde ressurstilgjengeligheten er null, mens den faktiske ressurstilgjengeligheten kan være tilstrekkelig til å levere de forespurte dataene. Med multinodestreaming

minimeres gapet mellom den opplevde og faktiske ressurstilgjengeligheten i nettverket. Ved å i tillegg utføre effektiv reetablering av en tapt strøm, kan dette gapet reduseres ytterligere.

6.5.1 Testresultatene

Gitt at den faktiske ressurstilgjengeligheten i P2P-nettverket er høyere enn antall ressursforespørsler (det vil si at det finnes et tilstrekkelig antall noder til å betjene samtlige objektforespørsler til det aktuelle objektet), vil forholdet mellom sannsynlighet for nodefeil, og degradert avspillingskvalitet som følge av en nodefeil, være et godt mål på gapet mellom faktisk- og opplevd ressurstilgjengelighet. Gjennom fem tester har vi beregnet den tiden det tar å reetablere en tapt strøm. I hver test har fire delstrømmer feilet, noe som har resultert i at fire delstrømmer har blitt reetablert per test.

Sannsynligheten for at en delstrøm feiler i løpet av avspillingstiden kan angis som antall feil per tidsenhet. I testene ble det generert fire feil per avspilling av et objekt med varighet på 80 sekunder. Konsekvensen av en tapt delstrøm (i de konkrete testene) er 25 % degradert avspillingskvalitet i gjennomsnittlig 4,7 sekunder (se vedlegg B). For våre tester kan den gjennomsnittlige avspillingsdegraderingen som følge av feil per test uttrykkes som $4{,}7s \cdot 25\ \% \cdot \frac{4}{80} = 5{,}9$ %. Selv med fire nodefeil i løpet av 80 sekunder er altså den totale kvalitetsdegraderingen på mindre enn 6 %. Samtidig har ikke kvalitetsdegraderingen på noe tidspunkt vært mer enn 25 %.

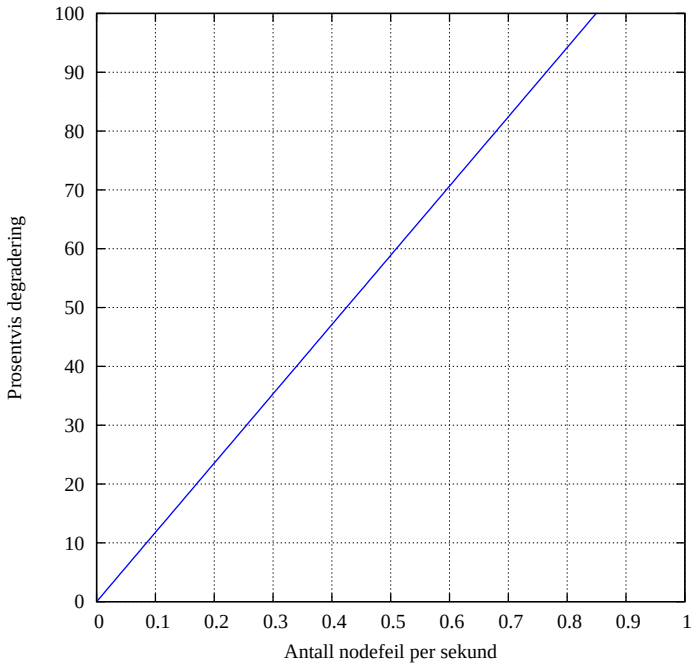
Uttrykket for total kvalitetsdegradering kan uttrykkes mer generelt som følger:

$$Degradering = t_r \cdot d \cdot f(t) \tag{6.1}$$

Hvor:

t_r: Tiden det tar å reetablere en tapt delstrøm.
d: Degraderingsfaktor (i forhold til full kvalitet) som følge av en tapt delstrøm.
$f(t)$: Delstrømfeil per tidsenhet

Grafen i figur 6.13 viser hvordan den totale kvalitetsdegraderingen varierer som funksjon av antall nodefeil per sekund. I grafen har vi brukt testresultatene fra kapittel 6.4.2 som mål for tiden det tar å gjenopprette en tapt delstrøm.



Figur 6.13: Avspillingsdegradering som funksjon av antall nodefeil per sekund.

Av figuren ser vi at ved for eksempel 0,5 delstrømfeil per sekund vil den totale avspillingsdegraderingen være omlag 60 %. Dette betyr imidlertid ikke at avspillingen vil kunne skje kontinuerlig med 40 % kvalitet fordi degraderingen er et gjennomsnitt for hele avspillingsperioden. I tilfeller hvor samtlige delstrømmer har feilet vil avspillingen stanse totalt for en liten periode, men dette vil ikke nødvendigvis ha store konsekvenser for målet for total avspillingsdegradering. Figur 6.13 gir derfor bare et tilnærmet bilde av avspillingskvaliteten slik brukeren ser det.

Ett av resultatene fra testene var at reetablering av en delstrøm gjennomsnittlig tok cirka 4,7 sekunder. Reetableringen skjer i følgende tre faser:

- Forhandling.
- Behandling av forhandlingsresultat.
- Etablering av strømmottak.

Gjennomsnittsvarigheten av hver fase var for de utførte testene henholdsvis 4,06, 0,54 og 0,11 sekunder (se vedlegg B). Forhandlingsfasen er den absolutt lengste fasen — faktisk utgjør den hele 86 % av tiden det tar å reetablere en strøm. I prototypen benyttet vi RMI for utførelsen av forhandlingsfasen. Det kan i ettertid diskuteres om RMI er den mest effektive måten å utføre en

såpass tidskritisk funksjon på. Hvis det er forbundet stor tilleggskostnad ved å sende enkle meldinger over nettverk ved hjelp av RMI, kan denne kostnaden ha urimelig stor påvirkning på metodens effektivitet. Ved å effektivisere meldingsutvekslingen i denne fasen, kan metoden potensielt kunne forbedres. Hvor stort dette forbedringspotensialet er, er det imidlertid vanskelig å anslå.

6.5.2 Testgjennomføringen

I tillegg til prototypen, kan selve gjennomføringen av testene påvirke testresultatet. I dette avsnittet vil vi forsøke å identifisere eventuelle feilkilder (i form av forenklinger, antagelser osv.) som har blitt gjort som kan påvirke utfallet av testene.

Det ble totalt gjennomført fem tester, og for hver test ble det generert feil for fire delstrømmer. Til sammen ble det altså gjort beregninger for 20 reetableringer. 20 målinger gir sammen et rimelig godt bilde på metodens effektivitet, men høyere nøyaktighet kan oppnås ved å utføre enda flere målinger.

Testene ble utført på NTNUs maskiner og nettverk. I selve P2P-nettverket var det 13 noder som alle var i besittelse av et replikat av det forespurte objektet. For testresultatet har omgivelsene hatt en klar betydning. Nettverksomgivelsene var gunstige og homogene samtidig som det var forholdsvis høy replikasjonsgrad. Egenskapene til maskinene som ble benyttet varierte imidlertid betydelig (med tanke på prosessorkraft, størrelsen og hastigheten på primærminnet og operativsystem). Fordi omgivelsene til et system vil kunne påvirke systemets funksjon og effektivitet, er realistiske omgivelser nødvendig. Vi er av den oppfatning at omgivelsene som ble benyttet i våre tester vil kunne være realistiske for mange applikasjoner, og at de derfor ikke har uforholdsmessig stor betydning for testresultatene. Andre omgivelser vil imidlertid kunne gi andre resultater, noe som er uunngåelig.

Konfigurasjonsparameterne i prototypen kan påvirke både dens funksjon og effektivitet. Hvis rammebetingelsene for prosjektet tillot at testing ble gjort i større skala, ville det være nyttig å utføre mange flere tester hvor konfigurasjonsparameterne ble variert. Andre verdier for disse parameterne ville kunnet påvirke utfallet av effektivitetsberegningene, og det ville derfor vært interessant å se om ytelsen til prototypen kunne forbedres ytterligere.

6.5.3 Evalueringskriteriene

Evalueringskriteriene for nodeadministrasjonsmetoden som ble fastsatt i kapittel 5.4, ble for tre av fem evalueringskriterier vurdert opp mot selve metodespesifikasjonen i kapittel 5.3. De resterende kriteriene var implementasjonsavhengige. Mer konkret gjenstår følgende ubesvarte evalueringskriterier:

5. Hvorvidt det er mulig å implementere metoden i et ekte P2P-system. Er løsningen implementerbar, og vil den fungere i eksisterende nettverks- og maskinvareomgivelser?

6. Gitt at metoden er implementerbar (det vil si at den funksjonelt sett er mulig å realisere): Hvor effektiv er metoden med tanke på å gjenopprette maksimal kvalitet (gitt at den faktiske ressurstilgjengeligheten i nettverket er tilstrekkelig for å utføre en slik gjenopprettelse). Hvor lang tid bruker hver fase i gjenopprettingen fra en feil er oppdaget til normaltilstand er gjenopprettet?

Ved å implementere og teste prototypen har vi blant annet demonstrert konseptet knyttet til multinodestreaming og reetablering av feilede delstrømmer. Vi har vist at metoden, som vi spesifiserte i kapittel 5, er implementerbar og at den fungerer under eksisterende nettverks- og maskinvareomgivelser. Vi har også vurdert metodens effektivitet under gitte forutsetninger som blant annet bestemmes av omgivelsene. Blant annet har vi testet prototypen under gunstige nettverksforhold. Samtidig hadde vi forholdsvis stor grad av ressursreplikasjon, noe som også påvirker metodens effektivitet. Fordi målet med nodeadministrasjonsmetoden er å utnytte ressurspotensialet i omgivelsene, er det metodens effektivitet i forhold til gitte omgivelser som er interessant. Vi har vist at metoden, gitt disse omgivelsene, har evnen til å reetablere en strøm i løpet av gjennomsnittlig 4,7 sekunder. Dette anser vi som rimelig raskt gitt at metoden fortsatt har et forbedringspotensiale i forhold til tilleggskostnad knyttet til RMI.

Kapittel 7

Diskusjon

I kapittel 5 rettet vi fokus på problemstillingene knyttet til tilgjengelighet og streaming av digitale medier i P2P-nettverk. I den forbindelse presenterte vi en nodeadministrasjonsmetode, som ved bruk av multinodestreaming har som mål å gjøre gapet mellom faktisk- og opplevd ressurstilgjengelighet så lite som mulig. I kapittel 6 presenterte vi en prototyp som implementerte den spesifiserte metoden. Testingen av prototypen ga viktige svar angående metodens funksjon og effekt, og på basis av testresultatene konkluderte vi med at metoden bidrar til å gjøre virkningen av varierende nodetilgjengelighet mindre kritisk i forhold til tjenestekvaliteten brukeren opplever. Effekten av den spesifiserte metoden er dermed i tråd med målene som ble satt for arbeidet med denne hovedoppgaven (se kapittel 1).

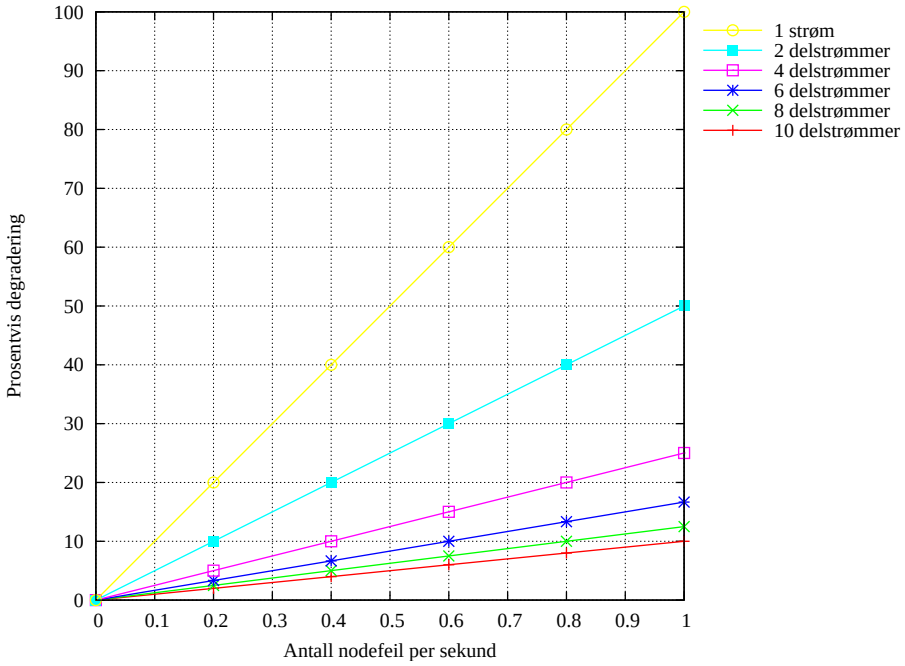
Det er flere problemstillinger knyttet til streaming i P2P-nettverk som er spesielt knyttet til multinodestreaming som teknikk. Flere av disse problemstillingene har blitt diskutert og svart på i metodeevalueringen i kapittel 5.4, og på bakgrunn av resultatene av testene i kapittel 6.4.2. Ved siden av disse problemstillingene er det er faktorer som er viktige ved realisering av et system i reelle omgivelser. I dette kapittelet vil vi vurdere og diskutere disse faktorene for å gi et helhetlig bilde av hvilke faktorer som er knyttet til realisering av et fullfunksjonelt system som ikke opererer under begrensningene og forenklingene som var tilfelle for prototypen.

7.1 Antall delstrømmer og reetableringstid

Testene på prototypen ble utført med medieobjekter bestående av fire delstrømmer. Antall delstrømmer som vi hadde muligheten til å benytte oss av var først og fremst begrenset av og det antall noder vi hadde tilgjengelig for testing. Avgjørelsen om å anvende fire delstrømmer var derfor snarere begrun-

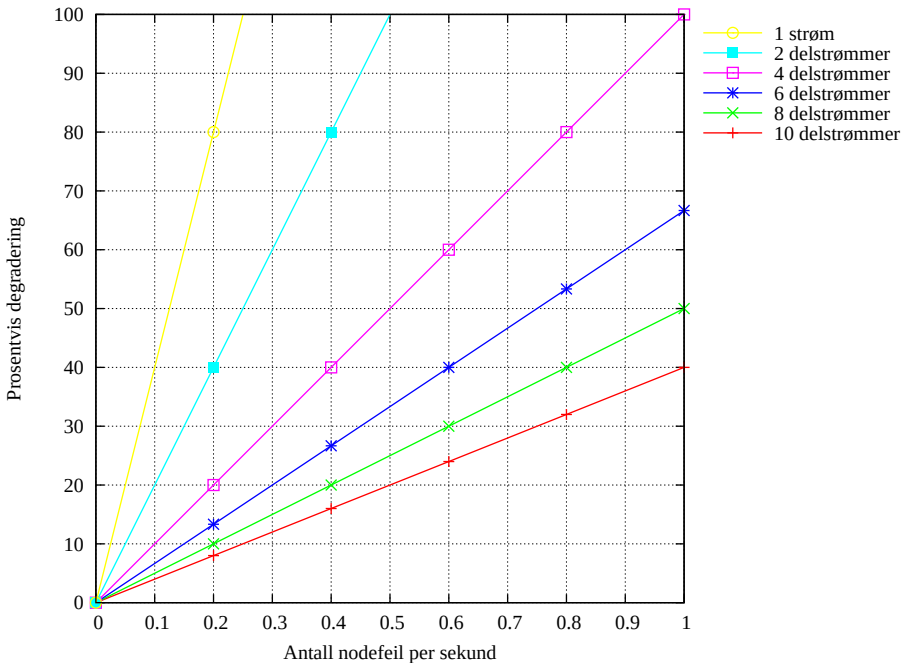
net ut fra vår tilgang til maskinressurser enn ut fra hva som faktisk er optimalt for multinodestreaming.

Figurene 7.1, 7.2 og 7.3 viser antall nodefeil per sekund i forhold til total prosentvis degradering (formel 6.1) med henholdsvis ett, fire og åtte sekunders reetableringstid. Ut fra figuren med fire sekunders reetableringstid ser vi at prosentvis degradering halveres når antall delstrømmer dobles. Dette stemmer overens med formel 6.1 som tilsier at total prosentvis degradering og degraderingsfaktor er proporsjonale størrelser.

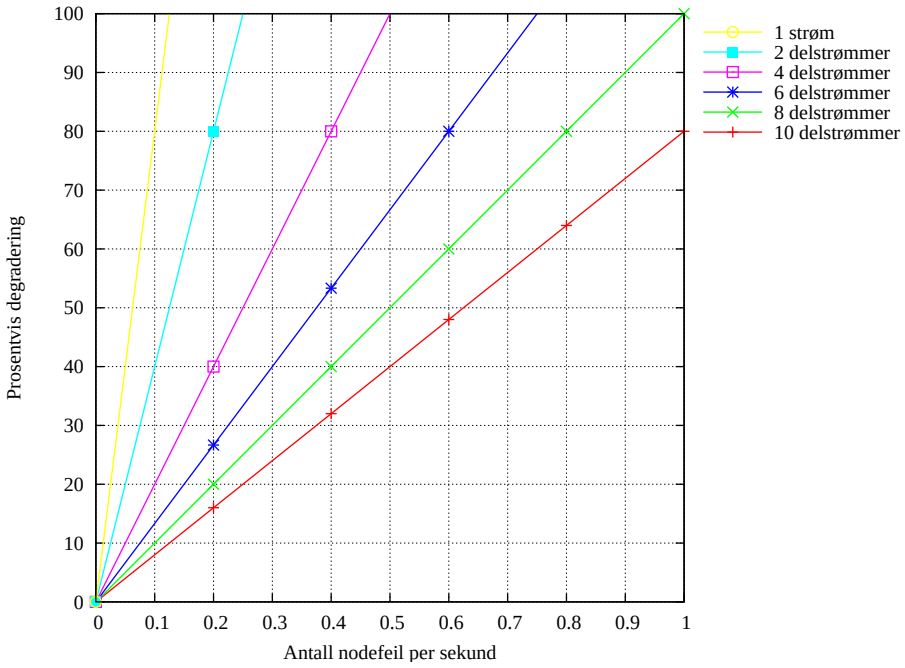


Figur 7.1: Avspillingsdegradering ved reetableringstid 1s.

I figur 7.1 og 7.3 ser vi at enkelte av grafene når 100 % degradering (langs andreaksen) før antall nodefeil per sekund (langs førsteaksen) når verdien én. Dette skyldes at reetableringstiden er lengre enn perioden mellom to nodefeil for et delobjekt, og etter at antall nodefeil per sekund når denne terskelverdien vil medieobjektet ikke lenger kunne spilles av. Sammenligner vi figur 7.1 med 7.3 ser vi at reetableringstiden har stor effekt på hvor mange nodefeil per sekund som kan tolereres. Dersom reetableringstiden er åtte sekunder, ser vi at metoden kun vil motta data for objekter delt i åtte eller ti delstrømmer etter at nodefeil per sekund overstiger en terskelverdi på omtrent 0,75.



Figur 7.2: Avspillingsdegradering ved reetableringstid 4s.



Figur 7.3: Avspillingsdegradering ved reetableringstid 8s.

Med en reetableringstid på ett sekund vil samtlige delstrømskonfigurasjoner kunne tolerere nodefeil per sekund opp til én. Et system med en lav reetableringstid kan med andre ord tolerere en høyere frekvens av nodefeil, og derfor bidrar en lav reetableringstid til at antall delstrømmer kan holdes lavt. I praksis vil terskelverdien for hvor stor prosentvis total degradering som kan tolereres selvsagt være langt lavere enn 100, men det gir likevel mening å sammenligne ekstremalverdiene. Reetableringstiden kan trygt sies å være den mest kritiske ytelsesparameteren til multinodestreaming. Det spesielle med reetableringstiden er at den er uavhengig av antall delstrømmer, og heller ikke har noen negativ virkning på de andre ytelsesparametrene til metoden. For å fastsette optimal reetableringstid er det derfor ikke nødvendig å avveie reetableringstid mot andre ytelsesparametre. Dette medfører at optimalisering av reetableringstid utelukkende har en positiv effekt på den totale degraderingsfaktoren.

Antall nodefeil per sekund som kan tolereres ved bruk av multinodestreaming øker med antall delstrømmer som et medieobjekt deles inn i. Ut fra figurene kan det derfor virke som om det optimale er å velge flest mulig delstrømmer for dermed å minimere den kvalitetsmessige degraderingsfaktoren i avspillingen som følge av en nodefeil. Figurene tar derimot ikke hensyn til at flere noder også vil medføre at flere nodefeil forekommer. Siden mottaker henter alle delstrømmer i et medieobjekt fra forskjellige avsendere, medfører oppdeling i flere delstrømmer at flere avsendernoder må være i besittelse av et replikat av medieobjektet. Dermed økes også kravet til replikasjonsgrad i P2P-systemet.

Leveranse fra flere noder medfører økt tilleggskostnad for mottak, synkronisering og dekomprimering hos mottakernoden. Forskjellen i ressursforbruk hos mottakernoden når én selvstendig strøm i forhold til fire delstrømmer ble mottatt var for vår prototyp neglisjerbar, men dette kan skyldes at målingene vi foretok var svært grovkornede og kun opererte med gjennomsnittsverdier over lengre tidsintervall. Dermed fikk vi ikke målt hvor ofte og hvor lenge ytelsesprofilen til de forskjellige formene for mottak nådde 100 % av den tilgjengelige prosessortiden. Avhengig av medieformatet som blir benyttet er det likevel trolig at mottakernoden vil belastes med større tilleggskostnader ved mottak av flere delstrømmer enn ved mottak av en helhetlig mediestrøm. For å bestemme antall delstrømmer bør følgende parametre vurderes opp mot hverandre:

- Antall nodefeil per sekund som skal tolereres.
- Antall tilgjengelige replikater.
- Reetableringstid.
- Degraderingsfaktor som følge av en nodefeil.
- Tilleggskostnad ved mottak, synkronisering og dekomprimering av en delstrøm.

Sammenlignet med tilfellet der medieobjektet kun består av én strøm, og følgelig også leveres fra en enkelt node, framstår tilfellet der medieobjektet deles inn i fire delstrømmer som et robust alternativ. I samtlige av de tre figurene tolererer multinodestreaming med fire delstrømmer vesentlig større feilrate enn hva tilfellet for enkeltnodestreaming gjør. Ut fra erfaringene med prototypen vet vi både at mottakernoder er i stand til å motta fire delstrømmer i MJPEG-format, og at det er fullt mulig å oppnå en reetableringstid rett i underkant av fem sekunder. Degraderingsfaktoren ved nodefeil når medieobjektet leveres som fire delstrømmer er 25 %, og det er mulig at dette vil oppleves som noe stort. Ut fra kvalitetsmessige hensyn bør derfor også tilfeller med flere enn fire delstrømmer vurderes. For fortsatt å holde antall noder som involveres for hver mottaker lavt framstår dermed seks delstrømmer som et alternativ. Vi mener derfor at det er realistisk å implementere et system for multinodestreaming der medieobjekter består av fire til seks delstrømmer.

7.2 Heterogene omgivelser

En utfordring ved realisering av streamingtjenester i P2P-nettverk er heterogene nettverksomgivelser. I kapittel 4.3 ble adaptive medieformater omtalt som en løsning for å kunne levere et objekt med kvalitet og båndbredde tilpasset den enkelte mottakernode. Dette behovet vil også være gjeldende i et P2P-nettverk som benytter multinodestreaming. Multinodestreaming krever at et medieobjekt deles i flere delobjekter slik at de forskjellige delene kan leveres fra ulike noder. Hvis omgivelsene ikke tillater at samtlige delstrømmer av et objekt leveres, kan en mottaker velge å bare motta en delmengde av delstrømmene. Med multinodestreaming er det altså allerede lagt til rette for adaptivitet så lenge medieformatet som brukes tillater adaptivitet i form delstrømmer som kan spilles av uavhengig av hverandre. I prototypen var konsekvensen av manglende delstrømmer at en hel kvadrant i avspillingsvinduet forsvant. Ved å bruke en mer sofistikert teknikk for å dele et objekt i flere delstrømmer, ville konsekvensen av adaptivitet kunne reduseres til for eksempel lavere oppløsning eller færre avspilte bilder per tidsenhet (se kapittel 4.3).

I heterogene nettverksomgivelser vil også nettverksbåndbredden ut fra den enkelte node variere. Noen noder vil være i stand til å levere data med høyere båndbredde enn andre. Multinodestreaming legger til rette for adaptivitet også for avsendernoden, noe som blant annet vil være ønskelig ved lav replikasjonsgrad i nettverket. Hvis det ikke er et tilstrekkelig antall noder i nettverket som er i besittelse av et bestemt objekt, vil det ikke være mulig å spre lasten for å levere objektet mellom like mange noder som det er delstrømmer. I slike tilfeller kan ressurssterke noder ta en større leveringsbyrde enn mindre ressurssterke noder ved at de leverer flere delobjekter til samme mottaker. Denne teknikken

kan brukes for å utnytte de tilgjengelige ressursene i nettverket bedre. I tilfeller hvor det er et tilstrekkelig antall noder til å levere ett delobjekt per node, er ikke adaptiv leveranse ønskelig fordi det øker konsekvensen av en nodefeil.

7.3 Distribusjon av delobjekter

En utfordring knyttet til multinodestreaming som vi ennå ikke har adressert er hvordan et medieobjekt deles og distribueres til andre noder i nettverket. Fordi multinodestreaming krever en viss replikasjonsgrad for å fungere optimalt, må systemet ha en felles løsning som sikrer at alle replikater av det samme objektet er identiske (slik at det er vilkårlig hvilket replikat som leveres), samtidig som at et objekt blir replikert etter hvert som objektet blir levert til nye noder.

For å generere delstrømmene til et objekt finnes to hovedalternativer:

- Generere delstrømmene ved kjøretid.
- Generere delstrømmene i forkant slik at de kan streames individuelt.

Generering av delstrømmer ved kjøretid krever at hver avsendernode må gjøre deling og omkoding av det opprinnelige objektet samtidig som den streamer data til mottakernoden. En slik løsning kan i utgangspunktet være gunstig fordi det muliggjør fleksibilitet med tanke på hvor mange delobjekter et objekt deles i. Hvis det for eksempel finnes fire tilgjengelige replikater av et objekt i P2P-nettverket, vil objektet kunne deles i fire deler. Finnes det flere replikater, vil antallet delobjekter kunne økes. Ulempen med en slik løsning er at splittingen av medieobjektet må gjøres hos hver enkelt avsendernode, noe som kan være forbundet med høye prosesseringskostnader.

Hvis delobjektene genereres én gang og deretter kan distribueres separat, vil antall delstrømmer et objekt leveres i være bestemt på forhånd. Fordelen med denne løsningen er at en avsendernode slipper å bruke prosesseringsressurser for å generere delstrømmer ved kjøretid. En potensiell utfordring knyttet til denne metoden er hvordan objektet skal leveres i nettverket når det kun finnes ett replikat. Multinodestreaming vil ikke være mulig å benytte i slike tilfeller. For å spre objektet initielt, må enkeltnodestreaming benyttes der én node leverer samtlige delobjekter. Denne fasen er altså kritisk for at objektet i det hele tatt får flere replikater i nettverket. Etter hvert som replikasjonsgraden øker, vil lasten kunne fordeles på et større antall noder. Et alternativ er å spre replikater av et objekt automatisk, for eksempel som en bakgrunnsprosess som eventuelt kan konfigureres til å bruke mindre båndbredde i forhold til om objektet skulle leveres i sanntid (for å ikke belaste en node unødig). Dermed vil antallet replikater av et objekt kunne økes uten at de streames som følge av en konkret objektforespørsel.

Kapittel 8

Konklusjon og videre arbeid

I denne hovedoppgaven har fokus vært å vurdere hvordan streaming av digitale medier kan utføres med tilfredsstillende tjenestekvalitet i P2P-nettverk. Et potensielt problem med streaming i slike systemer er at en node som leverer en strøm til en annen node kan bli utilgjengelig, noe som medfører avspillingsbrudd hos mottakeren av strømmen. Et av målene med oppgaven var å kartlegge egenskaper knyttet til P2P-nettverk som påvirker muligheten til å realisere streaming med tilfredsstillende tjenestekvalitet. Et videre mål var å spesifisere en metode som utnytter eksisterende egenskaper i P2P-nettverk for å minimere de tjenestekvalitetsmessige konsekvensene av varierende nodetilgjengelighet.

P2P og streaming er ingen enkel kombinasjon. Fordi P2P-nettverk består av noder på randen av Internett, med stor grad av selvstyre og med vidt forskjellige tekniske egenskaper, er omgivelsene for P2P-tjenester i stor grad preget av heterogenitet, dynamikk og dermed usikkerhet. P2P-systemer er usikre i den forstand at det er vanskelig å forutsi deres egenskaper. Et P2P-system vil endre seg over tid som følge av at deltakende noder kan melde seg inn og ut når som helst, og at det er stor variasjon i egenskapene til hver enkelt node. Leveranse av medier med sanntidsegenskaper har derfor vist seg å være en utfordring. En del av disse utfordringene kan håndteres ved bruk av tilpassede teknikker som muliggjør aktiviteter som til enhver tid er tilpasset egenskapene til systemet. Adaptivitet er en teknikk som bidrar til at den enkelte node kan tilpasse leveranse og mottak av ulike medier til ressurssituasjonen i nettverket. Når mange noder ønsker å få levert det samme objektet innen en viss tidsramme, og det ikke er et tilstrekkelig antall noder til å levere objektet med unicast-kommunikasjon, kan multicast i applikasjonslaget benyttes slik at mottakernodene videresender strømmen til andre noder.

Vårt konkrete bidrag i retning av å muliggjøre streaming i P2P-nettverk med forbedret tjenestekvalitet var en nodeadministrasjonsmetode som benytter mul-

tinodestreaming for å fordele ansvaret for å levere et medieobjekt mellom flere avsendernoder. Ved å fordele leveranseansvaret mellom flere noder, var målet å redusere konsekvensen av at en node ble utilgjengelig i løpet av en leveransesesjon. Utgangspunktet og forutsetningene for denne metoden var at medieobjekter eksisterer i flere replikater, og derfor kan leveres av flere noder. I tillegg til multinodestreaming, var en viktig funksjon i metoden å effektivt overlate ansvaret for leveranse av en tapt delstrøm til en alternativ leveransenode. Ved å kunne gjøre en effektiv leveranseovertakelse, vil ressurstilgjengeligheten, sett fra brukerens perspektiv, forbedres ytterligere. Hvorvidt metoden faktisk fungerte slik vi ønsket var vanskelig å vurdere ut fra spesifikasjonen alene. Det var imidlertid mulig å vurdere den opp mot andre kriterier som vi satte, og vi vurderte det slik at metoden var spesifisert i henhold til samtlige av disse kriteriene.

For å teste funksjonaliteten til nodeadministrasjonsmetoden, valgte vi å implementere den i en prototyp sammen med annen nødvendig funksjonalitet for å realisere et fullstendig P2P-system. Selv om prototypen opererte under enkelte begrensninger og forenklinger, var det mulig å besvare flere av problemstillingene knyttet til funksjon og effektivitet. Vi demonstrerte blant annet at metoden er implementerbar, og målte dens effekt på forholdet mellom faktisk- og opplevd nodetilgjengelighet i bestemte omgivelser hvor blant annet replikasjonsgraden var forholdsvis høy. Resultatene fra testene viste at den tjenestekvalitetsmessige degraderingen av en feilet delstrøm, under de forutsetningene som var tilfelle for prototypen, er omvendt proporsjonal med antall delstrømmer. Som forventet resulterte altså et større antall delstrømmer i lavere kvalitetsdegradering per feilede delstrøm. Reetableringen av en tapt strøm kunne gjøres på gjennomsnittlig 4,7 sekunder, med potensiale for forbedringer på grunn av antatt stor tilleggskostnad forbundet med forhandlingsfasene. Svarene fra prototyp-testene var i så måte svært positive med tanke på å sannsynliggjøre at multinodestreaming kan være en del av en løsning på problemene knyttet til tilgjengelighet/tjenestekvalitet.

Gjennom arbeidet med hovedoppgaven, og utviklingen og testingen av nodeadministrasjonsmetoden spesielt, har vi besvart problemstillingene som ble identifisert i kapittel 1. Vi har demonstrert at den opplevde tilgjengeligheten kan tilnærmes den faktiske ressurstilgjengeligheten i P2P-nettverket ved å spre leveringsansvaret mellom flere noder. Vi kan derfor konkludere med at de sentrale målene med hovedoppgaven er nådd, selv om en større tidsramme rundt prosjektet ville kunnet resultere i både dypere og bredere utforsking av domenet.

Etter vår oppfatning kan altså multinodestreaming være en del av svaret på sentrale utfordringer knyttet til tjenestekvalitet og streaming i P2P-nettverk. Multinodestreaming introduserer imidlertid også nye problemstillinger. Noen av disse ble til dels adressert i denne oppgaven, men flere åpne spørsmål gjen-

står for videre arbeid. Det er også andre temaer, som ikke er direkte knyttet til multinodestreaming, som ville vært interessante i en fortsettelse av vårt arbeid.

En av de viktigste direkte fortsettelsene på arbeidet fra denne hovedoppgaven ville vært å ta i bruk et medieformat som er tilrettelagt for leveranse av uavhengige delstrømmer. En nødvendig egenskap med et slikt medieformat er adaptivitet. I kapittel 4.3 vurderte vi en rekke teknikker som kan benyttes for å realisere et slikt medieformat, men presenterte ingen konkrete formater. I et videre arbeid ville valg av teknikk for å oppnå adaptivitet og etableringen av et konkret medieformat stått sentralt.

En begrensning som var aktuell under testingen av prototypen, var mangel på heterogene omgivelser. Fordi P2P-nettverk kjennetegnes nettopp av heterogenitet, ville det vært interessant å se hvordan multinodestreaming ville fungert under slike omgivelser. Vi har tidligere antatt at denne metoden er godt tilrettelagt for slike omgivelser, blant annet på grunn av at den benytter teknikker som også brukes for å oppnå adaptivitet, og det ville derfor vært nyttig å demonstrere at dette faktisk er tilfelle. Testing under heterogene omgivelser ville kanskje også fremprovosert feilsituasjoner som det ikke har vært mulig å oppnå under våre tester.

Andre begrensninger og forenklinger som ble gjort i forbindelse med prototypimplementasjonen (mulighet for strømovervåking, ressursreservasjon hos avsendernode, ikke-sentralisert overlagsnettverk og klassifisering av noder basert på nettverkstopologi), burde i forbindelse med en videreutvikling av prototypen vært realisert som faktiske egenskaper. Disse begrensningene har sannsynligvis ikke hatt stor innvirkning på våre resultater, men hadde tidsrammen for prosjektet tillatt det ville det vært en prioritert oppgave å vurdere hvordan disse kan realiseres.

I vårt arbeid har vi hatt stort fokus på multinodestreaming som teknikk for å nå målet om forbedret tjenestekvalitet. Selv om vårt fokus har vært orientert rundt spesifikasjonen og testingen av én teknikk, er vi åpne for at andre teknikker kan fungere som alternativ eller supplement til vår nodeadministrasjonsmetode. En løsning som vi ikke har omtalt i denne hovedoppgaven, men som potensielt kan ha positiv effekt i forhold til tilgjengelighet (både node- og ressurstilgjengelighet), er et belønningssystem som belønner noder som har høy tilgjengelighet. Årsaken til at denne metoden ikke har blitt vurdert som en parallell til multinodestreaming er at dens effekt ville være å forbedre nodetilgjengeligheten i stedet for å utnytte det eksisterende ressurspotensialet i P2P-nettverket. Et belønningssystem kan eksempelvis favorisere noder som har høy tilgjengelighet ved forespørsel av andres ressurser i nettverket. Dermed motiveres brukerne til å la deres ressurser være tilgjengelige for andre noder. Denne konkrete metoden vil kunne være et nyttig supplement til multinodestreaming,

og ville vært interessant å vurdere i et videre arbeid. Også *alternative* metoder ville vært interessante for videre arbeid, og i den forbindelse ville blant annet ytelsessimuleringer kunne gitt nyttige svar på de forskjellige metodenes fordeler og ulemper i forhold til hverandre.

Det er også andre veier vi kunne ønsket å gå i forbindelse med et videre arbeid. Fokus på gjenfinning, generering og behandling metadata og andre faktorer som påvirker kvaliteten på de tjenestene som tilbys av nettverket er interessante. Disse faktorene anser vi imidlertid for å ikke være direkte oppfølging av vårt arbeid, men snarere en alternativ retning å gå i forbindelse med utvikling av streamingtjenester for P2P-nettverk. I denne hovedoppgaven har vi fokusert på de aspektene av dette arbeidet som vi selv har vurdert som mest kritiske og interessante i forhold til målet om å forbedre brukerens opplevelse av tjenestekvalitet. Vår konklusjon er at P2P-arkitekturen gjør tjenester mer kompliserte og tvinger utvikleren til å tenke alternativt, men at den har så mange fordeler i forhold til tradisjonell klient/tjener-arkitektur at den bør og vil være i fokus under realiseringen av mange tjenester i framtiden.

Vedlegg A

Oppgavetekst

Utgangspunktet for denne hovedoppgaven var, som blant annet beskrevet i kapittel 1, et ønske om å gjøre streaming av digitale medier med tilfredsstillende kvalitet til en realitet i P2P-nettverk. Bakgrunnen for dette ønsket var at vi i fordypningsprosjektet[1], utført høsten 2003, gjorde en studie hvor lagringsteknologiske utfordringer og løsninger knyttet til videoleveranse var i fokus [Bjørnsen og Lohne, 2003]. I denne studien identifiserte vi flere problemer i form av dårlig skalerbarhet knyttet til videoleveranse i klient/tjener-baserte systemer. I vårt arbeid identifiserte vi blant annet P2P som et alternativ til klient/tjener-arkitekturen. Vi innså imidlertid at P2P-systemer er forbundet med andre utfordringer, spesielt knyttet tjenestekvalitet.

En naturlig fortsettelse på arbeidet fra fordypningsprosjektet var å vurdere om, og eventuelt hvordan, streamingtjenester kan realiseres i P2P-nettverk. Vi hadde imidlertid ikke oversikt over hvilke aspekter knyttet til P2P som representerte de største utfordringene. En viktig oppgave var derfor å kartlegge domenet, og heller identifisere problemstillinger etter hvert. Den opprinnelige oppgaveteksten var derfor ikke spesielt konkret, men var snarere en gjensidig forståelse mellom veileder og oss om at P2P var et komplekst domene som måtte utforskes i bredden først for å være i stand til å identifisere de viktigste utfordringene. Tittelen for prosjektet ble derfor satt til "Streaming i P2P-nettverk", mens selve oppgaveformuleringen gjenstod å utforme.

Fokus i prosjektet ble innsnevret etter hvert, og det ble tidlig klart at problemstillingene knyttet til tilgjengelighet var svært kritiske i forhold til å kunne realisere pålitelige streamingtjenester i P2P-nettverk. Den endelige oppgaveteksten, formulert av forfatterene selv som et resultat av en prosess i samarbeid med veileder Roger Midtstraum, er gjengitt nedenfor.

[1] TDT4740 Databaseteknikk og distribuerte systemer, fordypning

Tittel: Streaming i P2P-nettverk.

Oppgavetekst:
Streaming av digitale medier har tradisjonelt vært gjort i klient/tjener-baserte systemer hvor en eller flere ressurssterke tjenere har levert data til et stort antall ressurskonsumerende klienter. P2P-systemer skalerer generelt bedre enn klient/tjener-systemer fordi samtlige deltakende noder bidrar med ressurser i tillegg til å konsumere dem. P2P er derfor et potensielt alternativ for mange tjenestetyper, deriblant streaming av digitale medier som video og lyd. P2P-arkitekturen introduserer imidlertid nye utfordringer og problemstillinger. Et viktig fokus i oppgaven er å gjøre en vurdering av de viktigste utfordringene knyttet til realisering av streamingtjenester i P2P-nettverk. Videre vil det være sentralt å identifisere hva som er de største hinderne i forhold til å kunne tilby tjenestekvalitet i slike systemer, samt å bidra med en selvstendig løsning for en eller flere av de viktigste problemstillingene.

Vedlegg B

Testresultater

I dette kapittelet presenteres tallmaterialet som har blitt produsert i løpet av hver enkelt test av prototypen. For hver test ble følgende testdata registrert:

- Tidspunktet for alle hendelser som påvirker avspillingskvaliteten hos en mottakernode. For hver hendelse ble tidspunktet da hendelsen inntraff registrert, samt identifikatoren til delstrømmen hendelsen gjelder og antallet delstrømmer som mottas etter at hendelsen har inntruffet.
- Tidsforbruket for hver fase under reetablering av en delstrøm. Det ble utført individuelle målinger for hver feilet delstrøm og for hver test, til sammen 20 målinger.

Videre i dette kapittelet blir testresultatene for hver utført test gjengitt i sin helhet. Til slutt blir resultatene fra hver test oppsummert i form av gjennomsnittberegninger.

B.1 Test 1

Tabell B.1 viser hendelsesforløpet for test nummer 1.

Strøm-ID	Hendelse	Antall delstrømmer etter hendelse	Tidspunkt
	Start	0	-4.46
2	Ny	1	0.00
1	Ny	2	13.84
3	Ny	3	13.90
4	Ny	4	13.91
1	Tapt	3	18.85
1	Ny	4	22.17
2	Tapt	3	26.92
2	Ny	4	33.66
3	Tapt	3	39.33
3	Ny	4	43.58
4	Tapt	3	50.54
4	Ny	4	58.22
	Slutt	4	80.00

Tabell B.1: Hendelsesforløp for test 1.

Tabell B.2 viser tidsforbruket per fase under reetablering av en delstrøm for test nummer 1.

	Objekt 1	Objekt 2	Objekt 3	Objekt 4
Forhandling	2502	6889	3601	6401
Behandling av resultat	531	189	440	664
Etablering av strømmottak	90	143	113	439

Tabell B.2: Målt tidsforbruk (i millisekunder) for test nummer 1.

B.2 Test 2

Tabell B.3 viser hendelsesforløpet for test nummer 2.

Strøm-ID	Hendelse	Antall delstrømmer etter hendelse	Tidspunkt
	Start	0	-5.56
3	Ny	1	0.00
4	Ny	2	13.88
2	Ny	3	13.92
1	Ny	4	13.95
1	Tapt	3	19.07
1	Ny	4	22.33
2	Tapt	3	27.00
2	Ny	4	30.24
3	Tapt	3	36.51
3	Ny	4	39.8
4	Tapt	3	48.39
4	Ny	4	52.62
	Slutt	4	80.00

Tabell B.3: Hendelsesforløp for test 2.

Tabell B.4 viser tidsforbruket per fase under reetablering av en delstrøm for test nummer 2.

	Objekt 1	Objekt 2	Objekt 3	Objekt 4
Forhandling	2723	2455	2771	3455
Behandling av resultat	305	579	264	595
Etablering av strømmottak	82	88	100	111

Tabell B.4: Målt tidsforbruk (i millisekunder) for test nummer 2.

B.3 Test 3

Tabell B.5 viser hendelsesforløpet for test nummer 3.

Strøm-ID	Hendelse	Antall delstrømmer etter hendelse	Tidspunkt
	Start	0	-4.78
3	Ny	1	0.00
2	Ny	2	13.55
1	Ny	3	13.69
4	Ny	4	14.12
1	Tapt	3	18.42
1	Ny	4	22.72
2	Tapt	3	34.84
2	Ny	4	39.18
3	Tapt	3	50.30
3	Ny	4	54.53
4	Tapt	3	66.55
4	Ny	4	70.76
	Slutt	4	80.00

Tabell B.5: Hendelsesforløp for test 3.

Tabell B.6 viser tidsforbruket per fase under reetablering av en delstrøm for test nummer 3.

	Objekt 1	Objekt 2	Objekt 3	Objekt 4
Forhandling	3243	3429	3537	3143
Behandling av resultat	806	617	503	894
Etablering av strømmottak	114	104	67	68

Tabell B.6: Målt tidsforbruk (i millisekunder) for test nummer 3.

B.4 Test 4

Tabell B.7 viser hendelsesforløpet for test nummer 4.

Strøm-ID	Hendelse	Antall delstrømmer etter hendelse	Tidspunkt
	Start	0	-4.77
1	Ny	1	0.00
3	Ny	2	0.07
2	Ny	3	13.53
4	Ny	4	13.62
1	Tapt	3	19.42
1	Ny	4	22.78
2	Tapt	3	31.72
2	Ny	4	34.95
3	Tapt	3	44.74
3	Ny	4	49.96
4	Tapt	3	59.34
4	Ny	4	67.56
	Slutt	4	80.00

Tabell B.7: Hendelsesforløp for test 4.

Tabell B.8 viser tidsforbruket per fase under reetablering av en delstrøm for test nummer 4.

	Objekt 1	Objekt 2	Objekt 3	Objekt 4
Forhandling	2687	2442	4668	7406
Behandling av resultat	439	583	390	677
Etablering av strømmottak	77	81	50	53

Tabell B.8: Målt tidsforbruk (i millisekunder) for test nummer 4.

B.5 Test 5

Tabell B.9 viser hendelsesforløpet for test nummer 5.

Strøm-ID	Hendelse	Antall delstrømmer etter hendelse	Tidspunkt
	Start	0	-4.52
3	Ny	1	0.00
1	Ny	2	13.83
2	Ny	3	13.88
4	Ny	4	13.93
1	Tapt	3	19.16
1	Ny	4	24.49
2	Tapt	3	31.07
2	Ny	4	34.41
3	Tapt	3	40.09
3	Ny	4	43.32
4	Tapt	3	48.74
4	Ny	4	60.03
	Slutt	4	80.00

Tabell B.9: Hendelsesforløp for test 5.

Tabell B.10 viser tidsforbruket per fase under reetablering av en delstrøm for test nummer 5.

	Objekt 1	Objekt 2	Objekt 3	Objekt 4
Forhandling	4047	2511	2545	10747
Behandling av resultat	1009	523	484	373
Etablering av strømmottak	87	124	83	67

Tabell B.10: Målt tidsforbruk (i millisekunder) for test nummer 5.

B.6 Totalt resultat

Tabell B.11 viser gjennomsnittlig tidsforbruk for hver fase. Gjennomsnittet er beregnet på grunnlag av resultatene fra samtlige fem tester. Grunnlaget for hver fase er altså 20 feilsituasjoner (5 tester á 4 feil).

	Gjennomsnitt
Forhandling	4060
Behandling av resultat	543
Etablering av strømmottak	107

Tabell B.11: Gjennomsnittlig tidsforbruk (i millisekunder) for alle testene.

Den gjennomsnittlige tiden det har tatt å gjenopprette en delstrøm, fra delstrømfeilen blir oppdaget til en ny delstrøm er opprettet, er 4060 + 543 + 107 = 4710 ms $\approx$ 4,7 sekunder.

Ordforklaringer

AVI "Audio Video Interleave". Filformat for kontinuerlige medier som kan inneholde både lyd- og videostrømmer.

Demultiplekse Dele et sammensatt objekt i fragmenter. En mediestrøm kan for eksempel deles i én videostrøm og én lydstrøm.

Distribuert hashtabell En hashtabell som er fordelt utover flere noder.

Ekte P2P Et P2P-system uten noen for for sentral enhet eller funksjon.

Hashtabell En tabell som inneholder hashverdier. Gitt en hashverdi vil tabellen meget raskt kunne utføre et oppslag for å hente ut objektet som representeres av denne verdien.

Hashverdi En tilnærmet unik verdi beregnet ut fra et objekt sitt innhold eller verdi.

Hybrid P2P Et P2P-system som inneholder en eller annen form for sentral enhet eller funksjon.

Jitter Variasjon i forsinkelsen mellom to endepunkter ved datakommunikasjon.

Katalogtjeneste Tjeneste som tilbyr oppslag i en samling over tilgjengelige ressurser. I et P2P-system kan en katalogtjeneste for eksempel brukes for å lokalisere noder som tilbyr et spesielt objekt.

Kontinuerlig medium Et medium som inneholder en tidsdimensjon. Lyd og video er eksempler på kontinuerlige medier.

Kvantum Minste enhet som en mediestrøm er satt sammen av. I video er for eksempel et enkeltbilde et kvantum.

Mediedokument/-objekt/-strøm En helhetlig sammensetning av en eller flere kontinuerlige medier. Et mediedokument består kan for eksempel bestå av en videostrøm og en lydstrøm.

MJPEG "Motion JPEG". Variant av bildekompresjonsteknikken JPEG som anvendes for å komprimere video.

MPEG "Moving Picture Experts Group", standardiseringskomité for digitale medier. MPEG brukes også som fellesbetegnelse for samtlige standardformater som er publisert av denne komiteen.

Multicast Nettverkstrafikk der trafikken kringskastes til kun et utvalgt sett av mottakere.

Multinodestreaming Metode der en mottaker og flere avsender samarbeider om å levere én mediestrøm. Hver avsender leverer en delstrøm som av mottaker settes sammen til den opprinnelige mediestrømmen.

Multiplekse Sette sammen fragmenter til en helhet. En lydstrøm og en videostrøm kan for eksempel multiplekses til en sammensatt mediestrøm.

Node En datamaskin som deltar i et distribuert system, som for eksempel i et P2P-nettverk.

Overlagsnettverk Nettverkslag som i et P2P-system benyttes for å lokalisere ressurser og rute meldinger mellom deltakende noder.

Peer En node i et P2P-nettverk.

Peer-to-Peer (P2P) Et distribuert system der alle deltakende noder kommuniserer direkte med hverandre, og hver node både tilbyr og konsumerer ressurser.

Streaming Avspilling av kontinuerlige medier samtidig som de leveres over nettverk.

Synkronisere Samkjøring av tid. To medieobjekter spilles av synkronisert når begge befinner seg på samme sted i tidsdimensjonen og deretter beveger seg med samme hastighet langs tidsaksen.

TCP/IP "Transport Control Protocol"/"Internet Protocol". TCP/IP brukes som fellesbetegnelse for de mest brukte og standardiserte Internett-protokollene.

Unicast Datakommunikasjon med én avsender og én mottaker.

Video on Demand (VoD) En tjeneste som tilbyr visning av video på forespørsel.

Bibliografi

Apple Computer, Inc.
Movie trailers.
2004.
URL `http://www.apple.com/trailers/`.
Aksesstidspunkt: 21.05.2004.

Stig Inge Lea Bjørnsen og Erik Vihovde Lohne.
Videoleveranse: Lagringsteknologiske utfordringer og løsninger.
Fordypningsprosjekt, Norges teknisk-naturvitenskapelige universitet (NTNU), 2003.

Hector M. Briceño, Steven Gortler og Leonard McMillan.
NAIVE — Network aware internet video encoding.
I *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, side 251–260 (ACM Press, 1999).
ISBN 1-58113-151-8.

Robert Cailliau.
A little history of the world wide web.
World Wide Web Consortium, 2000.
URL `http://www.w3.org/History.html`.
Aksesstidspunkt: 06.02.2004.

Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron og Atul Singh.
Splitstream: High-bandwidth multicast in cooperative environments.
I *Proceedings of the nineteenth ACM symposium on Operating systems principles*, side 298–313 (ACM Press, 2003).
ISBN 1-58113-757-5.

I. Cidon, A. Khamisy og M. Sidi.
Analysis of packet loss processes in high-speed networks.
IEEE Transactions on Information Theory, bind 39, nr. 1, 1993.

D. Clark.

The design philosophy of the DARPA internet protocols.
I *Symposium proceedings on Communications architectures and protocols*, side 106–114 (ACM Press, 1988).
ISBN 0-89791-279-9.

Ian Clarke, Oskar Sandberg, Brandon Wiley og Theodore W. Hong.
Freenet: A distributed anonymous information storage and retrieval system.
Lecture Notes in Computer Science, bind 2009, side 46+, 2001.
ISSN: 0302-9743.

S. Deering og R. Hinden.
Internet Protocol, version 6 (IPv6) specification.
RFC 2460, IETF, 1998.

distributed.net.
distributed.net.
2004.
URL `http://distributed.net`.
Aksesstidspunkt: 04.02.2004.

Diego Doval og Donal O'Mahony.
Overlay networks: A scalable alternative for P2P.
IEEE Internet Computing, 2003.

FastTrack.
Fasttrack protocol.
2004.
URL `http://www.fact-index.com/f/fa/fasttrack.html`.
Aksesstidspunkt 18.02.2004.

D. J. Gemmel, H. M. Vin, D. D. Kandlur, P. Venkat Ramgan og L. A. Rowe.
Multimedia storage servers: A tutorial.
IEEE Computer, bind 28, nr. 5, side 40–49, 1995.

Gnutella.
RFC-Gnutella.
2004.
URL `http://rfc-gnutella.sourceforge.net/index.html`.
Aksesstidspunkt: 17.02.2004.

Vivek K. Goyal.
Multiple description coding: Compression meets the network.
IEEE Signal Processing Magazine, 2001.

Ze-Nian Li og Mark S. Drew.
Fundamentals of Multimedia (Prentice Hall, 2004), første utgave.
ISBN 0-13-061872-1.

Guolun Lu.
Multimedia Database Management Systems, kapittel 2, side 13–46 (Artec House, 1999).

Lyse AS.
Lyse AS: Bredbåndsprodukter.
2004.
URL `https://www.lyse.no/produkter/bredband/produkter/internett/`.
Aksesstidspunkt: 09.02.2004.

L. Melville, J. Walkerdine og I. Sommerville.
Report on the dependability properties of P2P architectures.
Teknisk rapport, Lancaster University, 2002.

Microsoft Corp.
DCOM technical overview.
White paper, 1996.
URL `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom/html/msdn_dcomtec.asp`.
Aksesstidspunkt: 18.02.2004.

Microsoft Corp.
Microsoft Netmeeting.
2004.
URL `http://www.microsoft.com/windows/NetMeeting/default.ASP`.
Aksesstidspunkt: 18.02.2004.

Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins og Zhichen Xu.
Peer-to-peer computing.
Teknisk rapport, HP Laboratories Palo Alto, 2002.

Nelson Minar og Marc Hedlund.
A network of peers: Peer-to-peer models through the history.
I Andy Oram (redaktør), *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, kapittel 1, side 3–20 (O'Reilly, 2001).

MPEG.
Moving Picture Experts Group (MPEG).
2004.
URL `http://www.chiariglione.org/mpeg/`.
Aksesstidspunkt: 09.03.2004.

napster.com.
napster.com.
2004.

URL http://www.napster.com.
Aksesstidspunkt: 09.02.2004.

OMG, Inc.
Common Object Request Broker Architecture (CORBA/IIOP).
Spesifikasjon, 2002.
URL http://www.omg.org/technology/documents/corba_spec_catalog.htm.
Aksesstidspunkt: 18.02.2004.

Project JXTA.
Project JXTA.
2004.
URL http://www.jxta.org.
Aksesstidspunkt: 17.02.2004.

Quazal Tech.
Net-z.
2004.
URL http://www.quazal.com/.
Aksesstidspunkt: 18.02.2004.

Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp og Scott Schenker.
A scalable content-addressable network.
I *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, side 161–172 (ACM Press, 2001).
ISBN 1-58113-411-8.

Matei Ripeanu.
Peer-to-peer architecture case study: Gnutella network.
Teknisk rapport, Computer Science Department, The University of Chicago, 2001.

Antony Rowstron og Peter Druschel.
Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems.
I *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, side 329–350 (2001).

Rüdiger Schollmeier.
A definition of peer-to-peer networking towards a delimitation against classical client-server concepts.
I *IFIP Workshop on IP and ATM Traffic Management* (Paris, Frankrike, 2001).

Rüdiger Schollmeier.
A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications.
I *First International Conference on Peer-to-Peer Computing (P2P'01)* (2002).

H. Schulzrinne, S. Casner, R. Frederick og V. Jacobson.
RTP: A transport protocol for real-time applications.
RFC 3550, IETF, 2003.

Seti@home.
Seti@home.
2004.
URL `http://setiathome.ssl.berkeley.edu`.
Aksesstidspunkt: 04.02.2004.

Clay Shirky.
Listening to napster.
I Andy Oram (redaktør), *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, kapittel 2, side 21–37 (O'Reilly, 2001).

Dinkar Sitaram og Asit Dan.
Multimedia Servers (Morgan Kaufmann Publishers, San Francisco, CA, 2000), første utgave.

Tyron Stading, Petros Maniatis og Mary Baker.
Peer-to-peer caching schemes to address flash crowds.
I Peter Druschel, Frans Kaashoek og Antony Rowstron (redaktører), *First International Workshop on Peer-to-Peer systems*, side 201–213. IPTPS (Springer-Verlag, 2002).
ISBN 3-540-44179-4.

Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek og Hari Balakrishnan.
Chord: A scalable peer-to-peer lookup service for internet applications.
I *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, side 149–160 (ACM Press, 2001).
ISBN 1-58113-411-8.

Sun Microsystems Inc.
Java Media Framework.
2004.
URL `http://java.sun.com/products/java-media/jmf/`.
Aksesstidspunkt: 19.05.2004.

Andrew S. Tanenbaum.
Computer Networks (Prentice Hall, 1996), tredje utgave.

Steve Waterhouse.
JXTA search: Distributed search for distributed networks.
White paper, Sun Microsystems, Inc., 2001.
URL `http://search.jxta.org/JXTAsearch.pdf`.
Aksesstidspunkt: 17.02.2004.

Ben Y. Zhao, John Kubiatowicz og Anthony D. Joseph.
Tapestry: An infrastructure for fault-resilient wide-area location and routing.
Teknisk rapport, University of California at Berkeley, 2001.

Weibin Zhao, David Olshefski og Henning Schulzrinne.
Internet quality of service: An overview.
Teknisk rapport CUCS-003-00, Columbia University, 2000.